Page 001 - Charter and Input Surface: Definition of the Operating Unit

TOPIC: [write the subject, domain, book, theory, project, or raw material here]
SUPREME AIM: [write what this HyperKB must enable, unlock, improve, or produce]
RAW MATERIAL: [paste sources, notes, outlines, constraints, prior drafts, data, or links here]

This manuscript is a copy-ready English HyperKB OS-Level production prompt. Only the three lines above are intentionally blank. Everything below is general, reusable, and designed to force very strong reasoning models to build operational knowledge bases rather than decorative long-form content.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Protect the first three input lines as the only blanks before expanding; never allow raw accumulation to masquerade as depth.
2. Make the mission explicit with explicit boundaries, confidence tags, and failure conditions.
3. Bind the output to operational usefulness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Declare that missing data becomes a gap log, not hallucinated content through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 002 - Charter and Input Surface: Scope Boundary and Non-Scope Guard

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is scope boundary and non-scope guard, which means the model must make the mission explicit, bind the output to operational usefulness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make the mission explicit before expanding; never allow raw accumulation to masquerade as depth.
2. Bind the output to operational usefulness with explicit boundaries, confidence tags, and failure conditions.
3. Declare that missing data becomes a gap log, not hallucinated content by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Protect the first three input lines as the only blanks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 003 - Charter and Input Surface: Kernel Extraction Move

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is kernel extraction move, which means the model must bind the output to operational usefulness, declare that missing data becomes a gap log, not hallucinated content, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Bind the output to operational usefulness before expanding; never allow raw accumulation to masquerade as depth.
2. Declare that missing data becomes a gap log, not hallucinated content with explicit boundaries, confidence tags, and failure conditions.
3. Protect the first three input lines as the only blanks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make the mission explicit through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 004 - Charter and Input Surface: Difference-Making Test

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is difference-making test, which means the model must declare that missing data becomes a gap log, not hallucinated content, protect the first three input lines as the only blanks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Declare that missing data becomes a gap log, not hallucinated content before expanding; never allow raw accumulation to masquerade as depth.
2. Protect the first three input lines as the only blanks with explicit boundaries, confidence tags, and failure conditions.
3. Make the mission explicit by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Bind the output to operational usefulness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 005 - Charter and Input Surface: Confidence and Evidence Tagging

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is confidence and evidence tagging, which means the model must protect the first three input lines as the only blanks, make the mission explicit, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Protect the first three input lines as the only blanks before expanding; never allow raw accumulation to masquerade as depth.
2. Make the mission explicit with explicit boundaries, confidence tags, and failure conditions.
3. Bind the output to operational usefulness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Declare that missing data becomes a gap log, not hallucinated content through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 006 - Charter and Input Surface: Concept Boundary Split

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is concept boundary split, which means the model must make the mission explicit, bind the output to operational usefulness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make the mission explicit before expanding; never allow raw accumulation to masquerade as depth.
2. Bind the output to operational usefulness with explicit boundaries, confidence tags, and failure conditions.
3. Declare that missing data becomes a gap log, not hallucinated content by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Protect the first three input lines as the only blanks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 007 - Charter and Input Surface: Problem Deepening Method

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is problem deepening method, which means the model must bind the output to operational usefulness, declare that missing data becomes a gap log, not hallucinated content, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Bind the output to operational usefulness before expanding; never allow raw accumulation to masquerade as depth.
2. Declare that missing data becomes a gap log, not hallucinated content with explicit boundaries, confidence tags, and failure conditions.
3. Protect the first three input lines as the only blanks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make the mission explicit through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 008 - Charter and Input Surface: Skill Conversion Move

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is skill conversion move, which means the model must declare that missing data becomes a gap log, not hallucinated content, protect the first three input lines as the only blanks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Declare that missing data becomes a gap log, not hallucinated content before expanding; never allow raw accumulation to masquerade as depth.
2. Protect the first three input lines as the only blanks with explicit boundaries, confidence tags, and failure conditions.
3. Make the mission explicit by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Bind the output to operational usefulness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 009 - Charter and Input Surface: Loop Wiring Move

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is loop wiring move, which means the model must protect the first three input lines as the only blanks, make the mission explicit, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Protect the first three input lines as the only blanks before expanding; never allow raw accumulation to masquerade as depth.
2. Make the mission explicit with explicit boundaries, confidence tags, and failure conditions.
3. Bind the output to operational usefulness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Declare that missing data becomes a gap log, not hallucinated content through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 010 - Charter and Input Surface: Tool Conversion Move

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is tool conversion move, which means the model must make the mission explicit, bind the output to operational usefulness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make the mission explicit before expanding; never allow raw accumulation to masquerade as depth.
2. Bind the output to operational usefulness with explicit boundaries, confidence tags, and failure conditions.
3. Declare that missing data becomes a gap log, not hallucinated content by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Protect the first three input lines as the only blanks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 011 - Charter and Input Surface: Retrieval and Metadata Move

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is retrieval and metadata move, which means the model must bind the output to operational usefulness, declare that missing data becomes a gap log, not hallucinated content, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Bind the output to operational usefulness before expanding; never allow raw accumulation to masquerade as depth.
2. Declare that missing data becomes a gap log, not hallucinated content with explicit boundaries, confidence tags, and failure conditions.
3. Protect the first three input lines as the only blanks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make the mission explicit through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 012 - Charter and Input Surface: Agent Role Assignment

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is agent role assignment, which means the model must declare that missing data becomes a gap log, not hallucinated content, protect the first three input lines as the only blanks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Declare that missing data becomes a gap log, not hallucinated content before expanding; never allow raw accumulation to masquerade as depth.
2. Protect the first three input lines as the only blanks with explicit boundaries, confidence tags, and failure conditions.
3. Make the mission explicit by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Bind the output to operational usefulness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 013 - Charter and Input Surface: Court of Destruction Pass

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is court of destruction pass, which means the model must protect the first three input lines as the only blanks, make the mission explicit, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Protect the first three input lines as the only blanks before expanding; never allow raw accumulation to masquerade as depth.
2. Make the mission explicit with explicit boundaries, confidence tags, and failure conditions.
3. Bind the output to operational usefulness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Declare that missing data becomes a gap log, not hallucinated content through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 014 - Charter and Input Surface: Repair and Reconstruction Pass

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is repair and reconstruction pass, which means the model must make the mission explicit, bind the output to operational usefulness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make the mission explicit before expanding; never allow raw accumulation to masquerade as depth.
2. Bind the output to operational usefulness with explicit boundaries, confidence tags, and failure conditions.
3. Declare that missing data becomes a gap log, not hallucinated content by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Protect the first three input lines as the only blanks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 015 - Charter and Input Surface: Training and Mastery Drill

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is training and mastery drill, which means the model must bind the output to operational usefulness, declare that missing data becomes a gap log, not hallucinated content, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Bind the output to operational usefulness before expanding; never allow raw accumulation to masquerade as depth.
2. Declare that missing data becomes a gap log, not hallucinated content with explicit boundaries, confidence tags, and failure conditions.
3. Protect the first three input lines as the only blanks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make the mission explicit through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 016 - Charter and Input Surface: Output Interface Contract

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is output interface contract, which means the model must declare that missing data becomes a gap log, not hallucinated content, protect the first three input lines as the only blanks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Declare that missing data becomes a gap log, not hallucinated content before expanding; never allow raw accumulation to masquerade as depth.
2. Protect the first three input lines as the only blanks with explicit boundaries, confidence tags, and failure conditions.
3. Make the mission explicit by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Bind the output to operational usefulness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 017 - Charter and Input Surface: Failure Mode Catalogue

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is failure mode catalogue, which means the model must protect the first three input lines as the only blanks, make the mission explicit, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Protect the first three input lines as the only blanks before expanding; never allow raw accumulation to masquerade as depth.
2. Make the mission explicit with explicit boundaries, confidence tags, and failure conditions.
3. Bind the output to operational usefulness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Declare that missing data becomes a gap log, not hallucinated content through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 018 - Charter and Input Surface: Update and Versioning Rule

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is update and versioning rule, which means the model must make the mission explicit, bind the output to operational usefulness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make the mission explicit before expanding; never allow raw accumulation to masquerade as depth.
2. Bind the output to operational usefulness with explicit boundaries, confidence tags, and failure conditions.
3. Declare that missing data becomes a gap log, not hallucinated content by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Protect the first three input lines as the only blanks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 019 - Charter and Input Surface: Novelty and Hypothesis Gate

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is novelty and hypothesis gate, which means the model must bind the output to operational usefulness, declare that missing data becomes a gap log, not hallucinated content, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Bind the output to operational usefulness before expanding; never allow raw accumulation to masquerade as depth.
2. Declare that missing data becomes a gap log, not hallucinated content with explicit boundaries, confidence tags, and failure conditions.
3. Protect the first three input lines as the only blanks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make the mission explicit through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 020 - Charter and Input Surface: Final Prompt Clause

This page belongs to Part 1: Charter and Input Surface. Its task is to turns any raw topic into an operating charter, a scoped mission, and a disciplined input surface. The local module is final prompt clause, which means the model must declare that missing data becomes a gap log, not hallucinated content, protect the first three input lines as the only blanks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Declare that missing data becomes a gap log, not hallucinated content before expanding; never allow raw accumulation to masquerade as depth.
2. Protect the first three input lines as the only blanks with explicit boundaries, confidence tags, and failure conditions.
3. Make the mission explicit by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Bind the output to operational usefulness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 021 - Knowledge Kernel Design: Definition of the Operating Unit

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is definition of the operating unit, which means the model must find the central question, name the hard knot, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Find the central question before expanding; never allow raw accumulation to masquerade as depth.
2. Name the hard knot with explicit boundaries, confidence tags, and failure conditions.
3. Extract founding distinctions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State the generative principle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 022 - Knowledge Kernel Design: Scope Boundary and Non-Scope Guard

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is scope boundary and non-scope guard, which means the model must name the hard knot, extract founding distinctions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name the hard knot before expanding; never allow raw accumulation to masquerade as depth.
2. Extract founding distinctions with explicit boundaries, confidence tags, and failure conditions.
3. State the generative principle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Find the central question through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 023 - Knowledge Kernel Design: Kernel Extraction Move

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is kernel extraction move, which means the model must extract founding distinctions, state the generative principle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract founding distinctions before expanding; never allow raw accumulation to masquerade as depth.
2. State the generative principle with explicit boundaries, confidence tags, and failure conditions.
3. Find the central question by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name the hard knot through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 024 - Knowledge Kernel Design: Difference-Making Test

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is difference-making test, which means the model must state the generative principle, find the central question, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State the generative principle before expanding; never allow raw accumulation to masquerade as depth.
2. Find the central question with explicit boundaries, confidence tags, and failure conditions.
3. Name the hard knot by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract founding distinctions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 025 - Knowledge Kernel Design: Confidence and Evidence Tagging

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is confidence and evidence tagging, which means the model must find the central question, name the hard knot, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Find the central question before expanding; never allow raw accumulation to masquerade as depth.
2. Name the hard knot with explicit boundaries, confidence tags, and failure conditions.
3. Extract founding distinctions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State the generative principle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 026 - Knowledge Kernel Design: Concept Boundary Split

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is concept boundary split, which means the model must name the hard knot, extract founding distinctions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name the hard knot before expanding; never allow raw accumulation to masquerade as depth.
2. Extract founding distinctions with explicit boundaries, confidence tags, and failure conditions.
3. State the generative principle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Find the central question through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 027 - Knowledge Kernel Design: Problem Deepening Method

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is problem deepening method, which means the model must extract founding distinctions, state the generative principle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract founding distinctions before expanding; never allow raw accumulation to masquerade as depth.
2. State the generative principle with explicit boundaries, confidence tags, and failure conditions.
3. Find the central question by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name the hard knot through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 028 - Knowledge Kernel Design: Skill Conversion Move

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is skill conversion move, which means the model must state the generative principle, find the central question, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State the generative principle before expanding; never allow raw accumulation to masquerade as depth.
2. Find the central question with explicit boundaries, confidence tags, and failure conditions.
3. Name the hard knot by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract founding distinctions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 029 - Knowledge Kernel Design: Loop Wiring Move

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is loop wiring move, which means the model must find the central question, name the hard knot, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Find the central question before expanding; never allow raw accumulation to masquerade as depth.
2. Name the hard knot with explicit boundaries, confidence tags, and failure conditions.
3. Extract founding distinctions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State the generative principle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 030 - Knowledge Kernel Design: Tool Conversion Move

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is tool conversion move, which means the model must name the hard knot, extract founding distinctions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name the hard knot before expanding; never allow raw accumulation to masquerade as depth.
2. Extract founding distinctions with explicit boundaries, confidence tags, and failure conditions.
3. State the generative principle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Find the central question through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 031 - Knowledge Kernel Design: Retrieval and Metadata Move

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is retrieval and metadata move, which means the model must extract founding distinctions, state the generative principle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract founding distinctions before expanding; never allow raw accumulation to masquerade as depth.
2. State the generative principle with explicit boundaries, confidence tags, and failure conditions.
3. Find the central question by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name the hard knot through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 032 - Knowledge Kernel Design: Agent Role Assignment

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is agent role assignment, which means the model must state the generative principle, find the central question, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State the generative principle before expanding; never allow raw accumulation to masquerade as depth.
2. Find the central question with explicit boundaries, confidence tags, and failure conditions.
3. Name the hard knot by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract founding distinctions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 033 - Knowledge Kernel Design: Court of Destruction Pass

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is court of destruction pass, which means the model must find the central question, name the hard knot, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Find the central question before expanding; never allow raw accumulation to masquerade as depth.
2. Name the hard knot with explicit boundaries, confidence tags, and failure conditions.
3. Extract founding distinctions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State the generative principle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 034 - Knowledge Kernel Design: Repair and Reconstruction Pass

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is repair and reconstruction pass, which means the model must name the hard knot, extract founding distinctions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name the hard knot before expanding; never allow raw accumulation to masquerade as depth.
2. Extract founding distinctions with explicit boundaries, confidence tags, and failure conditions.
3. State the generative principle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Find the central question through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 035 - Knowledge Kernel Design: Training and Mastery Drill

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is training and mastery drill, which means the model must extract founding distinctions, state the generative principle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract founding distinctions before expanding; never allow raw accumulation to masquerade as depth.
2. State the generative principle with explicit boundaries, confidence tags, and failure conditions.
3. Find the central question by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name the hard knot through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 036 - Knowledge Kernel Design: Output Interface Contract

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is output interface contract, which means the model must state the generative principle, find the central question, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State the generative principle before expanding; never allow raw accumulation to masquerade as depth.
2. Find the central question with explicit boundaries, confidence tags, and failure conditions.
3. Name the hard knot by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract founding distinctions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 037 - Knowledge Kernel Design: Failure Mode Catalogue

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is failure mode catalogue, which means the model must find the central question, name the hard knot, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Find the central question before expanding; never allow raw accumulation to masquerade as depth.
2. Name the hard knot with explicit boundaries, confidence tags, and failure conditions.
3. Extract founding distinctions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State the generative principle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 038 - Knowledge Kernel Design: Update and Versioning Rule

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is update and versioning rule, which means the model must name the hard knot, extract founding distinctions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name the hard knot before expanding; never allow raw accumulation to masquerade as depth.
2. Extract founding distinctions with explicit boundaries, confidence tags, and failure conditions.
3. State the generative principle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Find the central question through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 039 - Knowledge Kernel Design: Novelty and Hypothesis Gate

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is novelty and hypothesis gate, which means the model must extract founding distinctions, state the generative principle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract founding distinctions before expanding; never allow raw accumulation to masquerade as depth.
2. State the generative principle with explicit boundaries, confidence tags, and failure conditions.
3. Find the central question by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name the hard knot through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 040 - Knowledge Kernel Design: Final Prompt Clause

This page belongs to Part 2: Knowledge Kernel Design. Its task is to compresses the field into a generative kernel that can reconstruct the branches without becoming a shallow summary. The local module is final prompt clause, which means the model must state the generative principle, find the central question, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State the generative principle before expanding; never allow raw accumulation to masquerade as depth.
2. Find the central question with explicit boundaries, confidence tags, and failure conditions.
3. Name the hard knot by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract founding distinctions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 041 - MCD Difference Engine: Definition of the Operating Unit

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is definition of the operating unit, which means the model must score active difference, reject decorative information, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score active difference before expanding; never allow raw accumulation to masquerade as depth.
2. Reject decorative information with explicit boundaries, confidence tags, and failure conditions.
3. Rank centrality by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Convert differences into actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 042 - MCD Difference Engine: Scope Boundary and Non-Scope Guard

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is scope boundary and non-scope guard, which means the model must reject decorative information, rank centrality, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Reject decorative information before expanding; never allow raw accumulation to masquerade as depth.
2. Rank centrality with explicit boundaries, confidence tags, and failure conditions.
3. Convert differences into actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score active difference through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 043 - MCD Difference Engine: Kernel Extraction Move

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is kernel extraction move, which means the model must rank centrality, convert differences into actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank centrality before expanding; never allow raw accumulation to masquerade as depth.
2. Convert differences into actions with explicit boundaries, confidence tags, and failure conditions.
3. Score active difference by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Reject decorative information through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 044 - MCD Difference Engine: Difference-Making Test

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is difference-making test, which means the model must convert differences into actions, score active difference, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Convert differences into actions before expanding; never allow raw accumulation to masquerade as depth.
2. Score active difference with explicit boundaries, confidence tags, and failure conditions.
3. Reject decorative information by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank centrality through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 045 - MCD Difference Engine: Confidence and Evidence Tagging

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is confidence and evidence tagging, which means the model must score active difference, reject decorative information, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score active difference before expanding; never allow raw accumulation to masquerade as depth.
2. Reject decorative information with explicit boundaries, confidence tags, and failure conditions.
3. Rank centrality by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Convert differences into actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 046 - MCD Difference Engine: Concept Boundary Split

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is concept boundary split, which means the model must reject decorative information, rank centrality, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Reject decorative information before expanding; never allow raw accumulation to masquerade as depth.
2. Rank centrality with explicit boundaries, confidence tags, and failure conditions.
3. Convert differences into actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score active difference through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 047 - MCD Difference Engine: Problem Deepening Method

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is problem deepening method, which means the model must rank centrality, convert differences into actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank centrality before expanding; never allow raw accumulation to masquerade as depth.
2. Convert differences into actions with explicit boundaries, confidence tags, and failure conditions.
3. Score active difference by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Reject decorative information through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 048 - MCD Difference Engine: Skill Conversion Move

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is skill conversion move, which means the model must convert differences into actions, score active difference, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Convert differences into actions before expanding; never allow raw accumulation to masquerade as depth.
2. Score active difference with explicit boundaries, confidence tags, and failure conditions.
3. Reject decorative information by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank centrality through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 049 - MCD Difference Engine: Loop Wiring Move

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is loop wiring move, which means the model must score active difference, reject decorative information, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score active difference before expanding; never allow raw accumulation to masquerade as depth.
2. Reject decorative information with explicit boundaries, confidence tags, and failure conditions.
3. Rank centrality by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Convert differences into actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 050 - MCD Difference Engine: Tool Conversion Move

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is tool conversion move, which means the model must reject decorative information, rank centrality, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Reject decorative information before expanding; never allow raw accumulation to masquerade as depth.
2. Rank centrality with explicit boundaries, confidence tags, and failure conditions.
3. Convert differences into actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score active difference through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 051 - MCD Difference Engine: Retrieval and Metadata Move

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is retrieval and metadata move, which means the model must rank centrality, convert differences into actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank centrality before expanding; never allow raw accumulation to masquerade as depth.
2. Convert differences into actions with explicit boundaries, confidence tags, and failure conditions.
3. Score active difference by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Reject decorative information through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 052 - MCD Difference Engine: Agent Role Assignment

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is agent role assignment, which means the model must convert differences into actions, score active difference, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Convert differences into actions before expanding; never allow raw accumulation to masquerade as depth.
2. Score active difference with explicit boundaries, confidence tags, and failure conditions.
3. Reject decorative information by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank centrality through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 053 - MCD Difference Engine: Court of Destruction Pass

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is court of destruction pass, which means the model must score active difference, reject decorative information, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score active difference before expanding; never allow raw accumulation to masquerade as depth.
2. Reject decorative information with explicit boundaries, confidence tags, and failure conditions.
3. Rank centrality by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Convert differences into actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 054 - MCD Difference Engine: Repair and Reconstruction Pass

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is repair and reconstruction pass, which means the model must reject decorative information, rank centrality, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Reject decorative information before expanding; never allow raw accumulation to masquerade as depth.
2. Rank centrality with explicit boundaries, confidence tags, and failure conditions.
3. Convert differences into actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score active difference through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 055 - MCD Difference Engine: Training and Mastery Drill

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is training and mastery drill, which means the model must rank centrality, convert differences into actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank centrality before expanding; never allow raw accumulation to masquerade as depth.
2. Convert differences into actions with explicit boundaries, confidence tags, and failure conditions.
3. Score active difference by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Reject decorative information through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 056 - MCD Difference Engine: Output Interface Contract

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is output interface contract, which means the model must convert differences into actions, score active difference, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Convert differences into actions before expanding; never allow raw accumulation to masquerade as depth.
2. Score active difference with explicit boundaries, confidence tags, and failure conditions.
3. Reject decorative information by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank centrality through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 057 - MCD Difference Engine: Failure Mode Catalogue

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is failure mode catalogue, which means the model must score active difference, reject decorative information, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score active difference before expanding; never allow raw accumulation to masquerade as depth.
2. Reject decorative information with explicit boundaries, confidence tags, and failure conditions.
3. Rank centrality by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Convert differences into actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 058 - MCD Difference Engine: Update and Versioning Rule

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is update and versioning rule, which means the model must reject decorative information, rank centrality, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Reject decorative information before expanding; never allow raw accumulation to masquerade as depth.
2. Rank centrality with explicit boundaries, confidence tags, and failure conditions.
3. Convert differences into actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score active difference through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 059 - MCD Difference Engine: Novelty and Hypothesis Gate

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is novelty and hypothesis gate, which means the model must rank centrality, convert differences into actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank centrality before expanding; never allow raw accumulation to masquerade as depth.
2. Convert differences into actions with explicit boundaries, confidence tags, and failure conditions.
3. Score active difference by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Reject decorative information through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 060 - MCD Difference Engine: Final Prompt Clause

This page belongs to Part 3: MCD Difference Engine. Its task is to uses active difference as the value filter for concepts, claims, tools, and skills. The local module is final prompt clause, which means the model must convert differences into actions, score active difference, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Convert differences into actions before expanding; never allow raw accumulation to masquerade as depth.
2. Score active difference with explicit boundaries, confidence tags, and failure conditions.
3. Reject decorative information by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank centrality through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 061 - QMCD Measurement Discipline: Definition of the Operating Unit

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is definition of the operating unit, which means the model must use ordinal scales carefully, tag confidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Use ordinal scales carefully before expanding; never allow raw accumulation to masquerade as depth.
2. Tag confidence with explicit boundaries, confidence tags, and failure conditions.
3. Avoid fake precision by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 062 - QMCD Measurement Discipline: Scope Boundary and Non-Scope Guard

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is scope boundary and non-scope guard, which means the model must tag confidence, avoid fake precision, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag confidence before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid fake precision with explicit boundaries, confidence tags, and failure conditions.
3. Record uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Use ordinal scales carefully through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 063 - QMCD Measurement Discipline: Kernel Extraction Move

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is kernel extraction move, which means the model must avoid fake precision, record uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid fake precision before expanding; never allow raw accumulation to masquerade as depth.
2. Record uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Use ordinal scales carefully by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag confidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 064 - QMCD Measurement Discipline: Difference-Making Test

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is difference-making test, which means the model must record uncertainty, use ordinal scales carefully, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Use ordinal scales carefully with explicit boundaries, confidence tags, and failure conditions.
3. Tag confidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid fake precision through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 065 - QMCD Measurement Discipline: Confidence and Evidence Tagging

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is confidence and evidence tagging, which means the model must use ordinal scales carefully, tag confidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Use ordinal scales carefully before expanding; never allow raw accumulation to masquerade as depth.
2. Tag confidence with explicit boundaries, confidence tags, and failure conditions.
3. Avoid fake precision by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 066 - QMCD Measurement Discipline: Concept Boundary Split

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is concept boundary split, which means the model must tag confidence, avoid fake precision, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag confidence before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid fake precision with explicit boundaries, confidence tags, and failure conditions.
3. Record uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Use ordinal scales carefully through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 067 - QMCD Measurement Discipline: Problem Deepening Method

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is problem deepening method, which means the model must avoid fake precision, record uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid fake precision before expanding; never allow raw accumulation to masquerade as depth.
2. Record uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Use ordinal scales carefully by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag confidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 068 - QMCD Measurement Discipline: Skill Conversion Move

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is skill conversion move, which means the model must record uncertainty, use ordinal scales carefully, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Use ordinal scales carefully with explicit boundaries, confidence tags, and failure conditions.
3. Tag confidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid fake precision through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 069 - QMCD Measurement Discipline: Loop Wiring Move

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is loop wiring move, which means the model must use ordinal scales carefully, tag confidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Use ordinal scales carefully before expanding; never allow raw accumulation to masquerade as depth.
2. Tag confidence with explicit boundaries, confidence tags, and failure conditions.
3. Avoid fake precision by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 070 - QMCD Measurement Discipline: Tool Conversion Move

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is tool conversion move, which means the model must tag confidence, avoid fake precision, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag confidence before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid fake precision with explicit boundaries, confidence tags, and failure conditions.
3. Record uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Use ordinal scales carefully through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 071 - QMCD Measurement Discipline: Retrieval and Metadata Move

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is retrieval and metadata move, which means the model must avoid fake precision, record uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid fake precision before expanding; never allow raw accumulation to masquerade as depth.
2. Record uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Use ordinal scales carefully by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag confidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 072 - QMCD Measurement Discipline: Agent Role Assignment

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is agent role assignment, which means the model must record uncertainty, use ordinal scales carefully, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Use ordinal scales carefully with explicit boundaries, confidence tags, and failure conditions.
3. Tag confidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid fake precision through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 073 - QMCD Measurement Discipline: Court of Destruction Pass

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is court of destruction pass, which means the model must use ordinal scales carefully, tag confidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Use ordinal scales carefully before expanding; never allow raw accumulation to masquerade as depth.
2. Tag confidence with explicit boundaries, confidence tags, and failure conditions.
3. Avoid fake precision by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 074 - QMCD Measurement Discipline: Repair and Reconstruction Pass

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is repair and reconstruction pass, which means the model must tag confidence, avoid fake precision, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag confidence before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid fake precision with explicit boundaries, confidence tags, and failure conditions.
3. Record uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Use ordinal scales carefully through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 075 - QMCD Measurement Discipline: Training and Mastery Drill

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is training and mastery drill, which means the model must avoid fake precision, record uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid fake precision before expanding; never allow raw accumulation to masquerade as depth.
2. Record uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Use ordinal scales carefully by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag confidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 076 - QMCD Measurement Discipline: Output Interface Contract

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is output interface contract, which means the model must record uncertainty, use ordinal scales carefully, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Use ordinal scales carefully with explicit boundaries, confidence tags, and failure conditions.
3. Tag confidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid fake precision through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 077 - QMCD Measurement Discipline: Failure Mode Catalogue

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is failure mode catalogue, which means the model must use ordinal scales carefully, tag confidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Use ordinal scales carefully before expanding; never allow raw accumulation to masquerade as depth.
2. Tag confidence with explicit boundaries, confidence tags, and failure conditions.
3. Avoid fake precision by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 078 - QMCD Measurement Discipline: Update and Versioning Rule

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is update and versioning rule, which means the model must tag confidence, avoid fake precision, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag confidence before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid fake precision with explicit boundaries, confidence tags, and failure conditions.
3. Record uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Use ordinal scales carefully through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 079 - QMCD Measurement Discipline: Novelty and Hypothesis Gate

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is novelty and hypothesis gate, which means the model must avoid fake precision, record uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid fake precision before expanding; never allow raw accumulation to masquerade as depth.
2. Record uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Use ordinal scales carefully by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag confidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 080 - QMCD Measurement Discipline: Final Prompt Clause

This page belongs to Part 4: QMCD Measurement Discipline. Its task is to adds cautious scoring, confidence tags, centrality tags, and operational maturity levels without fake precision. The local module is final prompt clause, which means the model must record uncertainty, use ordinal scales carefully, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Use ordinal scales carefully with explicit boundaries, confidence tags, and failure conditions.
3. Tag confidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid fake precision through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 081 - MCAI Agency Intelligence: Definition of the Operating Unit

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is definition of the operating unit, which means the model must select actions, route cognitive resources, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Select actions before expanding; never allow raw accumulation to masquerade as depth.
2. Route cognitive resources with explicit boundaries, confidence tags, and failure conditions.
3. Model consequences by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Feed world effects back into memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 082 - MCAI Agency Intelligence: Scope Boundary and Non-Scope Guard

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is scope boundary and non-scope guard, which means the model must route cognitive resources, model consequences, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route cognitive resources before expanding; never allow raw accumulation to masquerade as depth.
2. Model consequences with explicit boundaries, confidence tags, and failure conditions.
3. Feed world effects back into memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Select actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 083 - MCAI Agency Intelligence: Kernel Extraction Move

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is kernel extraction move, which means the model must model consequences, feed world effects back into memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Model consequences before expanding; never allow raw accumulation to masquerade as depth.
2. Feed world effects back into memory with explicit boundaries, confidence tags, and failure conditions.
3. Select actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route cognitive resources through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 084 - MCAI Agency Intelligence: Difference-Making Test

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is difference-making test, which means the model must feed world effects back into memory, select actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Feed world effects back into memory before expanding; never allow raw accumulation to masquerade as depth.
2. Select actions with explicit boundaries, confidence tags, and failure conditions.
3. Route cognitive resources by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Model consequences through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 085 - MCAI Agency Intelligence: Confidence and Evidence Tagging

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is confidence and evidence tagging, which means the model must select actions, route cognitive resources, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Select actions before expanding; never allow raw accumulation to masquerade as depth.
2. Route cognitive resources with explicit boundaries, confidence tags, and failure conditions.
3. Model consequences by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Feed world effects back into memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 086 - MCAI Agency Intelligence: Concept Boundary Split

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is concept boundary split, which means the model must route cognitive resources, model consequences, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route cognitive resources before expanding; never allow raw accumulation to masquerade as depth.
2. Model consequences with explicit boundaries, confidence tags, and failure conditions.
3. Feed world effects back into memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Select actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 087 - MCAI Agency Intelligence: Problem Deepening Method

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is problem deepening method, which means the model must model consequences, feed world effects back into memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Model consequences before expanding; never allow raw accumulation to masquerade as depth.
2. Feed world effects back into memory with explicit boundaries, confidence tags, and failure conditions.
3. Select actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route cognitive resources through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 088 - MCAI Agency Intelligence: Skill Conversion Move

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is skill conversion move, which means the model must feed world effects back into memory, select actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Feed world effects back into memory before expanding; never allow raw accumulation to masquerade as depth.
2. Select actions with explicit boundaries, confidence tags, and failure conditions.
3. Route cognitive resources by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Model consequences through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 089 - MCAI Agency Intelligence: Loop Wiring Move

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is loop wiring move, which means the model must select actions, route cognitive resources, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Select actions before expanding; never allow raw accumulation to masquerade as depth.
2. Route cognitive resources with explicit boundaries, confidence tags, and failure conditions.
3. Model consequences by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Feed world effects back into memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 090 - MCAI Agency Intelligence: Tool Conversion Move

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is tool conversion move, which means the model must route cognitive resources, model consequences, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route cognitive resources before expanding; never allow raw accumulation to masquerade as depth.
2. Model consequences with explicit boundaries, confidence tags, and failure conditions.
3. Feed world effects back into memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Select actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 091 - MCAI Agency Intelligence: Retrieval and Metadata Move

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is retrieval and metadata move, which means the model must model consequences, feed world effects back into memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Model consequences before expanding; never allow raw accumulation to masquerade as depth.
2. Feed world effects back into memory with explicit boundaries, confidence tags, and failure conditions.
3. Select actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route cognitive resources through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 092 - MCAI Agency Intelligence: Agent Role Assignment

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is agent role assignment, which means the model must feed world effects back into memory, select actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Feed world effects back into memory before expanding; never allow raw accumulation to masquerade as depth.
2. Select actions with explicit boundaries, confidence tags, and failure conditions.
3. Route cognitive resources by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Model consequences through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 093 - MCAI Agency Intelligence: Court of Destruction Pass

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is court of destruction pass, which means the model must select actions, route cognitive resources, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Select actions before expanding; never allow raw accumulation to masquerade as depth.
2. Route cognitive resources with explicit boundaries, confidence tags, and failure conditions.
3. Model consequences by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Feed world effects back into memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 094 - MCAI Agency Intelligence: Repair and Reconstruction Pass

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is repair and reconstruction pass, which means the model must route cognitive resources, model consequences, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route cognitive resources before expanding; never allow raw accumulation to masquerade as depth.
2. Model consequences with explicit boundaries, confidence tags, and failure conditions.
3. Feed world effects back into memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Select actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 095 - MCAI Agency Intelligence: Training and Mastery Drill

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is training and mastery drill, which means the model must model consequences, feed world effects back into memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Model consequences before expanding; never allow raw accumulation to masquerade as depth.
2. Feed world effects back into memory with explicit boundaries, confidence tags, and failure conditions.
3. Select actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route cognitive resources through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 096 - MCAI Agency Intelligence: Output Interface Contract

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is output interface contract, which means the model must feed world effects back into memory, select actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Feed world effects back into memory before expanding; never allow raw accumulation to masquerade as depth.
2. Select actions with explicit boundaries, confidence tags, and failure conditions.
3. Route cognitive resources by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Model consequences through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 097 - MCAI Agency Intelligence: Failure Mode Catalogue

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is failure mode catalogue, which means the model must select actions, route cognitive resources, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Select actions before expanding; never allow raw accumulation to masquerade as depth.
2. Route cognitive resources with explicit boundaries, confidence tags, and failure conditions.
3. Model consequences by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Feed world effects back into memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 098 - MCAI Agency Intelligence: Update and Versioning Rule

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is update and versioning rule, which means the model must route cognitive resources, model consequences, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route cognitive resources before expanding; never allow raw accumulation to masquerade as depth.
2. Model consequences with explicit boundaries, confidence tags, and failure conditions.
3. Feed world effects back into memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Select actions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 099 - MCAI Agency Intelligence: Novelty and Hypothesis Gate

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is novelty and hypothesis gate, which means the model must model consequences, feed world effects back into memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Model consequences before expanding; never allow raw accumulation to masquerade as depth.
2. Feed world effects back into memory with explicit boundaries, confidence tags, and failure conditions.
3. Select actions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route cognitive resources through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 100 - MCAI Agency Intelligence: Final Prompt Clause

This page belongs to Part 5: MCAI Agency Intelligence. Its task is to converts knowledge into agency, decision, repair, action, and world-effect loops. The local module is final prompt clause, which means the model must feed world effects back into memory, select actions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Feed world effects back into memory before expanding; never allow raw accumulation to masquerade as depth.
2. Select actions with explicit boundaries, confidence tags, and failure conditions.
3. Route cognitive resources by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Model consequences through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 101 - HAKIM-MCD Module Stack: Definition of the Operating Unit

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is definition of the operating unit, which means the model must apply the extractor, draw the complexity map, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Apply the extractor before expanding; never allow raw accumulation to masquerade as depth.
2. Draw the complexity map with explicit boundaries, confidence tags, and failure conditions.
3. Run the court by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reconstruction memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 102 - HAKIM-MCD Module Stack: Scope Boundary and Non-Scope Guard

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is scope boundary and non-scope guard, which means the model must draw the complexity map, run the court, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Draw the complexity map before expanding; never allow raw accumulation to masquerade as depth.
2. Run the court with explicit boundaries, confidence tags, and failure conditions.
3. Write reconstruction memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Apply the extractor through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 103 - HAKIM-MCD Module Stack: Kernel Extraction Move

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is kernel extraction move, which means the model must run the court, write reconstruction memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run the court before expanding; never allow raw accumulation to masquerade as depth.
2. Write reconstruction memory with explicit boundaries, confidence tags, and failure conditions.
3. Apply the extractor by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Draw the complexity map through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 104 - HAKIM-MCD Module Stack: Difference-Making Test

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is difference-making test, which means the model must write reconstruction memory, apply the extractor, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reconstruction memory before expanding; never allow raw accumulation to masquerade as depth.
2. Apply the extractor with explicit boundaries, confidence tags, and failure conditions.
3. Draw the complexity map by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run the court through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 105 - HAKIM-MCD Module Stack: Confidence and Evidence Tagging

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is confidence and evidence tagging, which means the model must apply the extractor, draw the complexity map, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Apply the extractor before expanding; never allow raw accumulation to masquerade as depth.
2. Draw the complexity map with explicit boundaries, confidence tags, and failure conditions.
3. Run the court by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reconstruction memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 106 - HAKIM-MCD Module Stack: Concept Boundary Split

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is concept boundary split, which means the model must draw the complexity map, run the court, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Draw the complexity map before expanding; never allow raw accumulation to masquerade as depth.
2. Run the court with explicit boundaries, confidence tags, and failure conditions.
3. Write reconstruction memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Apply the extractor through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 107 - HAKIM-MCD Module Stack: Problem Deepening Method

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is problem deepening method, which means the model must run the court, write reconstruction memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run the court before expanding; never allow raw accumulation to masquerade as depth.
2. Write reconstruction memory with explicit boundaries, confidence tags, and failure conditions.
3. Apply the extractor by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Draw the complexity map through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 108 - HAKIM-MCD Module Stack: Skill Conversion Move

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is skill conversion move, which means the model must write reconstruction memory, apply the extractor, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reconstruction memory before expanding; never allow raw accumulation to masquerade as depth.
2. Apply the extractor with explicit boundaries, confidence tags, and failure conditions.
3. Draw the complexity map by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run the court through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 109 - HAKIM-MCD Module Stack: Loop Wiring Move

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is loop wiring move, which means the model must apply the extractor, draw the complexity map, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Apply the extractor before expanding; never allow raw accumulation to masquerade as depth.
2. Draw the complexity map with explicit boundaries, confidence tags, and failure conditions.
3. Run the court by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reconstruction memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 110 - HAKIM-MCD Module Stack: Tool Conversion Move

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is tool conversion move, which means the model must draw the complexity map, run the court, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Draw the complexity map before expanding; never allow raw accumulation to masquerade as depth.
2. Run the court with explicit boundaries, confidence tags, and failure conditions.
3. Write reconstruction memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Apply the extractor through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 111 - HAKIM-MCD Module Stack: Retrieval and Metadata Move

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is retrieval and metadata move, which means the model must run the court, write reconstruction memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run the court before expanding; never allow raw accumulation to masquerade as depth.
2. Write reconstruction memory with explicit boundaries, confidence tags, and failure conditions.
3. Apply the extractor by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Draw the complexity map through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 112 - HAKIM-MCD Module Stack: Agent Role Assignment

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is agent role assignment, which means the model must write reconstruction memory, apply the extractor, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reconstruction memory before expanding; never allow raw accumulation to masquerade as depth.
2. Apply the extractor with explicit boundaries, confidence tags, and failure conditions.
3. Draw the complexity map by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run the court through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 113 - HAKIM-MCD Module Stack: Court of Destruction Pass

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is court of destruction pass, which means the model must apply the extractor, draw the complexity map, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Apply the extractor before expanding; never allow raw accumulation to masquerade as depth.
2. Draw the complexity map with explicit boundaries, confidence tags, and failure conditions.
3. Run the court by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reconstruction memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 114 - HAKIM-MCD Module Stack: Repair and Reconstruction Pass

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is repair and reconstruction pass, which means the model must draw the complexity map, run the court, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Draw the complexity map before expanding; never allow raw accumulation to masquerade as depth.
2. Run the court with explicit boundaries, confidence tags, and failure conditions.
3. Write reconstruction memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Apply the extractor through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 115 - HAKIM-MCD Module Stack: Training and Mastery Drill

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is training and mastery drill, which means the model must run the court, write reconstruction memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run the court before expanding; never allow raw accumulation to masquerade as depth.
2. Write reconstruction memory with explicit boundaries, confidence tags, and failure conditions.
3. Apply the extractor by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Draw the complexity map through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 116 - HAKIM-MCD Module Stack: Output Interface Contract

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is output interface contract, which means the model must write reconstruction memory, apply the extractor, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reconstruction memory before expanding; never allow raw accumulation to masquerade as depth.
2. Apply the extractor with explicit boundaries, confidence tags, and failure conditions.
3. Draw the complexity map by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run the court through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 117 - HAKIM-MCD Module Stack: Failure Mode Catalogue

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is failure mode catalogue, which means the model must apply the extractor, draw the complexity map, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Apply the extractor before expanding; never allow raw accumulation to masquerade as depth.
2. Draw the complexity map with explicit boundaries, confidence tags, and failure conditions.
3. Run the court by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reconstruction memory through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 118 - HAKIM-MCD Module Stack: Update and Versioning Rule

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is update and versioning rule, which means the model must draw the complexity map, run the court, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Draw the complexity map before expanding; never allow raw accumulation to masquerade as depth.
2. Run the court with explicit boundaries, confidence tags, and failure conditions.
3. Write reconstruction memory by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Apply the extractor through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 119 - HAKIM-MCD Module Stack: Novelty and Hypothesis Gate

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is novelty and hypothesis gate, which means the model must run the court, write reconstruction memory, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run the court before expanding; never allow raw accumulation to masquerade as depth.
2. Write reconstruction memory with explicit boundaries, confidence tags, and failure conditions.
3. Apply the extractor by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Draw the complexity map through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 120 - HAKIM-MCD Module Stack: Final Prompt Clause

This page belongs to Part 6: HAKIM-MCD Module Stack. Its task is to structures the knowledge base through extractor, mapper, router, court, loop, and reconstruction memory modules. The local module is final prompt clause, which means the model must write reconstruction memory, apply the extractor, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reconstruction memory before expanding; never allow raw accumulation to masquerade as depth.
2. Apply the extractor with explicit boundaries, confidence tags, and failure conditions.
3. Draw the complexity map by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run the court through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 121 - Ingestion and Source Governance: Definition of the Operating Unit

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is definition of the operating unit, which means the model must classify source type, separate original from secondary, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must convert the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify source type before expanding; never allow raw accumulation to masquerade as depth.
2. Separate original from secondary with explicit boundaries, confidence tags, and failure conditions.
3. Track provenance by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark unsupported synthesis through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 122 - Ingestion and Source Governance: Scope Boundary and Non-Scope Guard

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is scope boundary and non-scope guard, which means the model must separate original from secondary, track provenance, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must test the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate original from secondary before expanding; never allow raw accumulation to masquerade as depth.
2. Track provenance with explicit boundaries, confidence tags, and failure conditions.
3. Mark unsupported synthesis by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify source type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 123 - Ingestion and Source Governance: Kernel Extraction Move

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is kernel extraction move, which means the model must track provenance, mark unsupported synthesis, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must repair the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track provenance before expanding; never allow raw accumulation to masquerade as depth.
2. Mark unsupported synthesis with explicit boundaries, confidence tags, and failure conditions.
3. Classify source type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate original from secondary through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 124 - Ingestion and Source Governance: Difference-Making Test

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is difference-making test, which means the model must mark unsupported synthesis, classify source type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must upgrade the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark unsupported synthesis before expanding; never allow raw accumulation to masquerade as depth.
2. Classify source type with explicit boundaries, confidence tags, and failure conditions.
3. Separate original from secondary by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track provenance through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 125 - Ingestion and Source Governance: Confidence and Evidence Tagging

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is confidence and evidence tagging, which means the model must classify source type, separate original from secondary, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must index the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify source type before expanding; never allow raw accumulation to masquerade as depth.
2. Separate original from secondary with explicit boundaries, confidence tags, and failure conditions.
3. Track provenance by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark unsupported synthesis through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 126 - Ingestion and Source Governance: Concept Boundary Split

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is concept boundary split, which means the model must separate original from secondary, track provenance, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must train the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate original from secondary before expanding; never allow raw accumulation to masquerade as depth.
2. Track provenance with explicit boundaries, confidence tags, and failure conditions.
3. Mark unsupported synthesis by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify source type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 127 - Ingestion and Source Governance: Problem Deepening Method

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is problem deepening method, which means the model must track provenance, mark unsupported synthesis, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must extract the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track provenance before expanding; never allow raw accumulation to masquerade as depth.
2. Mark unsupported synthesis with explicit boundaries, confidence tags, and failure conditions.
3. Classify source type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate original from secondary through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 128 - Ingestion and Source Governance: Skill Conversion Move

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is skill conversion move, which means the model must mark unsupported synthesis, classify source type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must compress the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark unsupported synthesis before expanding; never allow raw accumulation to masquerade as depth.
2. Classify source type with explicit boundaries, confidence tags, and failure conditions.
3. Separate original from secondary by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track provenance through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 129 - Ingestion and Source Governance: Loop Wiring Move

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is loop wiring move, which means the model must classify source type, separate original from secondary, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must map the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify source type before expanding; never allow raw accumulation to masquerade as depth.
2. Separate original from secondary with explicit boundaries, confidence tags, and failure conditions.
3. Track provenance by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark unsupported synthesis through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 130 - Ingestion and Source Governance: Tool Conversion Move

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is tool conversion move, which means the model must separate original from secondary, track provenance, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must route the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate original from secondary before expanding; never allow raw accumulation to masquerade as depth.
2. Track provenance with explicit boundaries, confidence tags, and failure conditions.
3. Mark unsupported synthesis by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify source type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 131 - Ingestion and Source Governance: Retrieval and Metadata Move

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is retrieval and metadata move, which means the model must track provenance, mark unsupported synthesis, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must score the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track provenance before expanding; never allow raw accumulation to masquerade as depth.
2. Mark unsupported synthesis with explicit boundaries, confidence tags, and failure conditions.
3. Classify source type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate original from secondary through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 132 - Ingestion and Source Governance: Agent Role Assignment

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is agent role assignment, which means the model must mark unsupported synthesis, classify source type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must separate the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark unsupported synthesis before expanding; never allow raw accumulation to masquerade as depth.
2. Classify source type with explicit boundaries, confidence tags, and failure conditions.
3. Separate original from secondary by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track provenance through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 133 - Ingestion and Source Governance: Court of Destruction Pass

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is court of destruction pass, which means the model must classify source type, separate original from secondary, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must convert the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify source type before expanding; never allow raw accumulation to masquerade as depth.
2. Separate original from secondary with explicit boundaries, confidence tags, and failure conditions.
3. Track provenance by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark unsupported synthesis through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 134 - Ingestion and Source Governance: Repair and Reconstruction Pass

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is repair and reconstruction pass, which means the model must separate original from secondary, track provenance, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must test the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate original from secondary before expanding; never allow raw accumulation to masquerade as depth.
2. Track provenance with explicit boundaries, confidence tags, and failure conditions.
3. Mark unsupported synthesis by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify source type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 135 - Ingestion and Source Governance: Training and Mastery Drill

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is training and mastery drill, which means the model must track provenance, mark unsupported synthesis, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must repair the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track provenance before expanding; never allow raw accumulation to masquerade as depth.
2. Mark unsupported synthesis with explicit boundaries, confidence tags, and failure conditions.
3. Classify source type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate original from secondary through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 136 - Ingestion and Source Governance: Output Interface Contract

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is output interface contract, which means the model must mark unsupported synthesis, classify source type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must upgrade the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark unsupported synthesis before expanding; never allow raw accumulation to masquerade as depth.
2. Classify source type with explicit boundaries, confidence tags, and failure conditions.
3. Separate original from secondary by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track provenance through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 137 - Ingestion and Source Governance: Failure Mode Catalogue

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is failure mode catalogue, which means the model must classify source type, separate original from secondary, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must index the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify source type before expanding; never allow raw accumulation to masquerade as depth.
2. Separate original from secondary with explicit boundaries, confidence tags, and failure conditions.
3. Track provenance by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark unsupported synthesis through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 138 - Ingestion and Source Governance: Update and Versioning Rule

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is update and versioning rule, which means the model must separate original from secondary, track provenance, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must train the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate original from secondary before expanding; never allow raw accumulation to masquerade as depth.
2. Track provenance with explicit boundaries, confidence tags, and failure conditions.
3. Mark unsupported synthesis by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify source type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 139 - Ingestion and Source Governance: Novelty and Hypothesis Gate

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is novelty and hypothesis gate, which means the model must track provenance, mark unsupported synthesis, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must extract the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track provenance before expanding; never allow raw accumulation to masquerade as depth.
2. Mark unsupported synthesis with explicit boundaries, confidence tags, and failure conditions.
3. Classify source type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate original from secondary through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 140 - Ingestion and Source Governance: Final Prompt Clause

This page belongs to Part 7: Ingestion and Source Governance. Its task is to controls how texts, notes, data, books, links, and prior conversations enter the base without corrupting trust. The local module is final prompt clause, which means the model must mark unsupported synthesis, classify source type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must compress the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark unsupported synthesis before expanding; never allow raw accumulation to masquerade as depth.
2. Classify source type with explicit boundaries, confidence tags, and failure conditions.
3. Separate original from secondary by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track provenance through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 141 - Concept Engineering Layer: Definition of the Operating Unit

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is definition of the operating unit, which means the model must define functionally, split near-synonyms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must test the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define functionally before expanding; never allow raw accumulation to masquerade as depth.
2. Split near-synonyms with explicit boundaries, confidence tags, and failure conditions.
3. Map confusions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test with examples through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 142 - Concept Engineering Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is scope boundary and non-scope guard, which means the model must split near-synonyms, map confusions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must repair the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Split near-synonyms before expanding; never allow raw accumulation to masquerade as depth.
2. Map confusions with explicit boundaries, confidence tags, and failure conditions.
3. Test with examples by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define functionally through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 143 - Concept Engineering Layer: Kernel Extraction Move

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is kernel extraction move, which means the model must map confusions, test with examples, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must upgrade the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map confusions before expanding; never allow raw accumulation to masquerade as depth.
2. Test with examples with explicit boundaries, confidence tags, and failure conditions.
3. Define functionally by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Split near-synonyms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 144 - Concept Engineering Layer: Difference-Making Test

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is difference-making test, which means the model must test with examples, define functionally, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must index the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test with examples before expanding; never allow raw accumulation to masquerade as depth.
2. Define functionally with explicit boundaries, confidence tags, and failure conditions.
3. Split near-synonyms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map confusions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 145 - Concept Engineering Layer: Confidence and Evidence Tagging

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is confidence and evidence tagging, which means the model must define functionally, split near-synonyms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must train the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define functionally before expanding; never allow raw accumulation to masquerade as depth.
2. Split near-synonyms with explicit boundaries, confidence tags, and failure conditions.
3. Map confusions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test with examples through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 146 - Concept Engineering Layer: Concept Boundary Split

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is concept boundary split, which means the model must split near-synonyms, map confusions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must extract the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Split near-synonyms before expanding; never allow raw accumulation to masquerade as depth.
2. Map confusions with explicit boundaries, confidence tags, and failure conditions.
3. Test with examples by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define functionally through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 147 - Concept Engineering Layer: Problem Deepening Method

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is problem deepening method, which means the model must map confusions, test with examples, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must compress the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map confusions before expanding; never allow raw accumulation to masquerade as depth.
2. Test with examples with explicit boundaries, confidence tags, and failure conditions.
3. Define functionally by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Split near-synonyms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 148 - Concept Engineering Layer: Skill Conversion Move

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is skill conversion move, which means the model must test with examples, define functionally, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must map the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test with examples before expanding; never allow raw accumulation to masquerade as depth.
2. Define functionally with explicit boundaries, confidence tags, and failure conditions.
3. Split near-synonyms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map confusions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 149 - Concept Engineering Layer: Loop Wiring Move

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is loop wiring move, which means the model must define functionally, split near-synonyms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must route the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define functionally before expanding; never allow raw accumulation to masquerade as depth.
2. Split near-synonyms with explicit boundaries, confidence tags, and failure conditions.
3. Map confusions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test with examples through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 150 - Concept Engineering Layer: Tool Conversion Move

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is tool conversion move, which means the model must split near-synonyms, map confusions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must score the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Split near-synonyms before expanding; never allow raw accumulation to masquerade as depth.
2. Map confusions with explicit boundaries, confidence tags, and failure conditions.
3. Test with examples by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define functionally through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 151 - Concept Engineering Layer: Retrieval and Metadata Move

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is retrieval and metadata move, which means the model must map confusions, test with examples, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must separate the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map confusions before expanding; never allow raw accumulation to masquerade as depth.
2. Test with examples with explicit boundaries, confidence tags, and failure conditions.
3. Define functionally by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Split near-synonyms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 152 - Concept Engineering Layer: Agent Role Assignment

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is agent role assignment, which means the model must test with examples, define functionally, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must convert the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test with examples before expanding; never allow raw accumulation to masquerade as depth.
2. Define functionally with explicit boundaries, confidence tags, and failure conditions.
3. Split near-synonyms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map confusions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 153 - Concept Engineering Layer: Court of Destruction Pass

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is court of destruction pass, which means the model must define functionally, split near-synonyms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must test the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define functionally before expanding; never allow raw accumulation to masquerade as depth.
2. Split near-synonyms with explicit boundaries, confidence tags, and failure conditions.
3. Map confusions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test with examples through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 154 - Concept Engineering Layer: Repair and Reconstruction Pass

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is repair and reconstruction pass, which means the model must split near-synonyms, map confusions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must repair the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Split near-synonyms before expanding; never allow raw accumulation to masquerade as depth.
2. Map confusions with explicit boundaries, confidence tags, and failure conditions.
3. Test with examples by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define functionally through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 155 - Concept Engineering Layer: Training and Mastery Drill

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is training and mastery drill, which means the model must map confusions, test with examples, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must upgrade the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map confusions before expanding; never allow raw accumulation to masquerade as depth.
2. Test with examples with explicit boundaries, confidence tags, and failure conditions.
3. Define functionally by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Split near-synonyms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 156 - Concept Engineering Layer: Output Interface Contract

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is output interface contract, which means the model must test with examples, define functionally, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must index the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test with examples before expanding; never allow raw accumulation to masquerade as depth.
2. Define functionally with explicit boundaries, confidence tags, and failure conditions.
3. Split near-synonyms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map confusions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 157 - Concept Engineering Layer: Failure Mode Catalogue

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is failure mode catalogue, which means the model must define functionally, split near-synonyms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must train the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define functionally before expanding; never allow raw accumulation to masquerade as depth.
2. Split near-synonyms with explicit boundaries, confidence tags, and failure conditions.
3. Map confusions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test with examples through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 158 - Concept Engineering Layer: Update and Versioning Rule

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is update and versioning rule, which means the model must split near-synonyms, map confusions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must extract the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Split near-synonyms before expanding; never allow raw accumulation to masquerade as depth.
2. Map confusions with explicit boundaries, confidence tags, and failure conditions.
3. Test with examples by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define functionally through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 159 - Concept Engineering Layer: Novelty and Hypothesis Gate

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is novelty and hypothesis gate, which means the model must map confusions, test with examples, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must compress the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map confusions before expanding; never allow raw accumulation to masquerade as depth.
2. Test with examples with explicit boundaries, confidence tags, and failure conditions.
3. Define functionally by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Split near-synonyms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 160 - Concept Engineering Layer: Final Prompt Clause

This page belongs to Part 8: Concept Engineering Layer. Its task is to turns terms into functional concepts, distinctions, boundaries, and anti-confusion devices. The local module is final prompt clause, which means the model must test with examples, define functionally, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must map the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test with examples before expanding; never allow raw accumulation to masquerade as depth.
2. Define functionally with explicit boundaries, confidence tags, and failure conditions.
3. Split near-synonyms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map confusions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 161 - Evidence and Confidence Layer: Definition of the Operating Unit

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is definition of the operating unit, which means the model must separate documented from plausible, preserve controversy, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate documented from plausible before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve controversy with explicit boundaries, confidence tags, and failure conditions.
3. State gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Downgrade overclaims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 162 - Evidence and Confidence Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is scope boundary and non-scope guard, which means the model must preserve controversy, state gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve controversy before expanding; never allow raw accumulation to masquerade as depth.
2. State gaps with explicit boundaries, confidence tags, and failure conditions.
3. Downgrade overclaims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate documented from plausible through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 163 - Evidence and Confidence Layer: Kernel Extraction Move

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is kernel extraction move, which means the model must state gaps, downgrade overclaims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Downgrade overclaims with explicit boundaries, confidence tags, and failure conditions.
3. Separate documented from plausible by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve controversy through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 164 - Evidence and Confidence Layer: Difference-Making Test

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is difference-making test, which means the model must downgrade overclaims, separate documented from plausible, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Downgrade overclaims before expanding; never allow raw accumulation to masquerade as depth.
2. Separate documented from plausible with explicit boundaries, confidence tags, and failure conditions.
3. Preserve controversy by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 165 - Evidence and Confidence Layer: Confidence and Evidence Tagging

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is confidence and evidence tagging, which means the model must separate documented from plausible, preserve controversy, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate documented from plausible before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve controversy with explicit boundaries, confidence tags, and failure conditions.
3. State gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Downgrade overclaims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 166 - Evidence and Confidence Layer: Concept Boundary Split

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is concept boundary split, which means the model must preserve controversy, state gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve controversy before expanding; never allow raw accumulation to masquerade as depth.
2. State gaps with explicit boundaries, confidence tags, and failure conditions.
3. Downgrade overclaims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate documented from plausible through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 167 - Evidence and Confidence Layer: Problem Deepening Method

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is problem deepening method, which means the model must state gaps, downgrade overclaims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Downgrade overclaims with explicit boundaries, confidence tags, and failure conditions.
3. Separate documented from plausible by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve controversy through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 168 - Evidence and Confidence Layer: Skill Conversion Move

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is skill conversion move, which means the model must downgrade overclaims, separate documented from plausible, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Downgrade overclaims before expanding; never allow raw accumulation to masquerade as depth.
2. Separate documented from plausible with explicit boundaries, confidence tags, and failure conditions.
3. Preserve controversy by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 169 - Evidence and Confidence Layer: Loop Wiring Move

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is loop wiring move, which means the model must separate documented from plausible, preserve controversy, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate documented from plausible before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve controversy with explicit boundaries, confidence tags, and failure conditions.
3. State gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Downgrade overclaims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 170 - Evidence and Confidence Layer: Tool Conversion Move

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is tool conversion move, which means the model must preserve controversy, state gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve controversy before expanding; never allow raw accumulation to masquerade as depth.
2. State gaps with explicit boundaries, confidence tags, and failure conditions.
3. Downgrade overclaims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate documented from plausible through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 171 - Evidence and Confidence Layer: Retrieval and Metadata Move

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is retrieval and metadata move, which means the model must state gaps, downgrade overclaims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Downgrade overclaims with explicit boundaries, confidence tags, and failure conditions.
3. Separate documented from plausible by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve controversy through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 172 - Evidence and Confidence Layer: Agent Role Assignment

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is agent role assignment, which means the model must downgrade overclaims, separate documented from plausible, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Downgrade overclaims before expanding; never allow raw accumulation to masquerade as depth.
2. Separate documented from plausible with explicit boundaries, confidence tags, and failure conditions.
3. Preserve controversy by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 173 - Evidence and Confidence Layer: Court of Destruction Pass

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is court of destruction pass, which means the model must separate documented from plausible, preserve controversy, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate documented from plausible before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve controversy with explicit boundaries, confidence tags, and failure conditions.
3. State gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Downgrade overclaims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 174 - Evidence and Confidence Layer: Repair and Reconstruction Pass

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is repair and reconstruction pass, which means the model must preserve controversy, state gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve controversy before expanding; never allow raw accumulation to masquerade as depth.
2. State gaps with explicit boundaries, confidence tags, and failure conditions.
3. Downgrade overclaims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate documented from plausible through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 175 - Evidence and Confidence Layer: Training and Mastery Drill

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is training and mastery drill, which means the model must state gaps, downgrade overclaims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Downgrade overclaims with explicit boundaries, confidence tags, and failure conditions.
3. Separate documented from plausible by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve controversy through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 176 - Evidence and Confidence Layer: Output Interface Contract

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is output interface contract, which means the model must downgrade overclaims, separate documented from plausible, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Downgrade overclaims before expanding; never allow raw accumulation to masquerade as depth.
2. Separate documented from plausible with explicit boundaries, confidence tags, and failure conditions.
3. Preserve controversy by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 177 - Evidence and Confidence Layer: Failure Mode Catalogue

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is failure mode catalogue, which means the model must separate documented from plausible, preserve controversy, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate documented from plausible before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve controversy with explicit boundaries, confidence tags, and failure conditions.
3. State gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Downgrade overclaims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 178 - Evidence and Confidence Layer: Update and Versioning Rule

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is update and versioning rule, which means the model must preserve controversy, state gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve controversy before expanding; never allow raw accumulation to masquerade as depth.
2. State gaps with explicit boundaries, confidence tags, and failure conditions.
3. Downgrade overclaims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate documented from plausible through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 179 - Evidence and Confidence Layer: Novelty and Hypothesis Gate

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is novelty and hypothesis gate, which means the model must state gaps, downgrade overclaims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Downgrade overclaims with explicit boundaries, confidence tags, and failure conditions.
3. Separate documented from plausible by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve controversy through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 180 - Evidence and Confidence Layer: Final Prompt Clause

This page belongs to Part 9: Evidence and Confidence Layer. Its task is to separates documented knowledge, plausible synthesis, controversy, hypothesis, and unknowns. The local module is final prompt clause, which means the model must downgrade overclaims, separate documented from plausible, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Downgrade overclaims before expanding; never allow raw accumulation to masquerade as depth.
2. Separate documented from plausible with explicit boundaries, confidence tags, and failure conditions.
3. Preserve controversy by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 181 - Problem Cartography Layer: Definition of the Operating Unit

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is definition of the operating unit, which means the model must surface hidden problems, distinguish symptoms from root knots, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Surface hidden problems before expanding; never allow raw accumulation to masquerade as depth.
2. Distinguish symptoms from root knots with explicit boundaries, confidence tags, and failure conditions.
3. Map stakeholders or schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define tests of solution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 182 - Problem Cartography Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is scope boundary and non-scope guard, which means the model must distinguish symptoms from root knots, map stakeholders or schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Distinguish symptoms from root knots before expanding; never allow raw accumulation to masquerade as depth.
2. Map stakeholders or schools with explicit boundaries, confidence tags, and failure conditions.
3. Define tests of solution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Surface hidden problems through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 183 - Problem Cartography Layer: Kernel Extraction Move

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is kernel extraction move, which means the model must map stakeholders or schools, define tests of solution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map stakeholders or schools before expanding; never allow raw accumulation to masquerade as depth.
2. Define tests of solution with explicit boundaries, confidence tags, and failure conditions.
3. Surface hidden problems by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Distinguish symptoms from root knots through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 184 - Problem Cartography Layer: Difference-Making Test

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is difference-making test, which means the model must define tests of solution, surface hidden problems, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define tests of solution before expanding; never allow raw accumulation to masquerade as depth.
2. Surface hidden problems with explicit boundaries, confidence tags, and failure conditions.
3. Distinguish symptoms from root knots by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map stakeholders or schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 185 - Problem Cartography Layer: Confidence and Evidence Tagging

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is confidence and evidence tagging, which means the model must surface hidden problems, distinguish symptoms from root knots, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Surface hidden problems before expanding; never allow raw accumulation to masquerade as depth.
2. Distinguish symptoms from root knots with explicit boundaries, confidence tags, and failure conditions.
3. Map stakeholders or schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define tests of solution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 186 - Problem Cartography Layer: Concept Boundary Split

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is concept boundary split, which means the model must distinguish symptoms from root knots, map stakeholders or schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Distinguish symptoms from root knots before expanding; never allow raw accumulation to masquerade as depth.
2. Map stakeholders or schools with explicit boundaries, confidence tags, and failure conditions.
3. Define tests of solution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Surface hidden problems through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 187 - Problem Cartography Layer: Problem Deepening Method

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is problem deepening method, which means the model must map stakeholders or schools, define tests of solution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map stakeholders or schools before expanding; never allow raw accumulation to masquerade as depth.
2. Define tests of solution with explicit boundaries, confidence tags, and failure conditions.
3. Surface hidden problems by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Distinguish symptoms from root knots through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 188 - Problem Cartography Layer: Skill Conversion Move

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is skill conversion move, which means the model must define tests of solution, surface hidden problems, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define tests of solution before expanding; never allow raw accumulation to masquerade as depth.
2. Surface hidden problems with explicit boundaries, confidence tags, and failure conditions.
3. Distinguish symptoms from root knots by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map stakeholders or schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 189 - Problem Cartography Layer: Loop Wiring Move

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is loop wiring move, which means the model must surface hidden problems, distinguish symptoms from root knots, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Surface hidden problems before expanding; never allow raw accumulation to masquerade as depth.
2. Distinguish symptoms from root knots with explicit boundaries, confidence tags, and failure conditions.
3. Map stakeholders or schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define tests of solution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 190 - Problem Cartography Layer: Tool Conversion Move

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is tool conversion move, which means the model must distinguish symptoms from root knots, map stakeholders or schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Distinguish symptoms from root knots before expanding; never allow raw accumulation to masquerade as depth.
2. Map stakeholders or schools with explicit boundaries, confidence tags, and failure conditions.
3. Define tests of solution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Surface hidden problems through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 191 - Problem Cartography Layer: Retrieval and Metadata Move

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is retrieval and metadata move, which means the model must map stakeholders or schools, define tests of solution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map stakeholders or schools before expanding; never allow raw accumulation to masquerade as depth.
2. Define tests of solution with explicit boundaries, confidence tags, and failure conditions.
3. Surface hidden problems by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Distinguish symptoms from root knots through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 192 - Problem Cartography Layer: Agent Role Assignment

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is agent role assignment, which means the model must define tests of solution, surface hidden problems, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define tests of solution before expanding; never allow raw accumulation to masquerade as depth.
2. Surface hidden problems with explicit boundaries, confidence tags, and failure conditions.
3. Distinguish symptoms from root knots by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map stakeholders or schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 193 - Problem Cartography Layer: Court of Destruction Pass

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is court of destruction pass, which means the model must surface hidden problems, distinguish symptoms from root knots, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Surface hidden problems before expanding; never allow raw accumulation to masquerade as depth.
2. Distinguish symptoms from root knots with explicit boundaries, confidence tags, and failure conditions.
3. Map stakeholders or schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define tests of solution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 194 - Problem Cartography Layer: Repair and Reconstruction Pass

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is repair and reconstruction pass, which means the model must distinguish symptoms from root knots, map stakeholders or schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Distinguish symptoms from root knots before expanding; never allow raw accumulation to masquerade as depth.
2. Map stakeholders or schools with explicit boundaries, confidence tags, and failure conditions.
3. Define tests of solution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Surface hidden problems through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 195 - Problem Cartography Layer: Training and Mastery Drill

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is training and mastery drill, which means the model must map stakeholders or schools, define tests of solution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map stakeholders or schools before expanding; never allow raw accumulation to masquerade as depth.
2. Define tests of solution with explicit boundaries, confidence tags, and failure conditions.
3. Surface hidden problems by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Distinguish symptoms from root knots through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 196 - Problem Cartography Layer: Output Interface Contract

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is output interface contract, which means the model must define tests of solution, surface hidden problems, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define tests of solution before expanding; never allow raw accumulation to masquerade as depth.
2. Surface hidden problems with explicit boundaries, confidence tags, and failure conditions.
3. Distinguish symptoms from root knots by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map stakeholders or schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 197 - Problem Cartography Layer: Failure Mode Catalogue

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is failure mode catalogue, which means the model must surface hidden problems, distinguish symptoms from root knots, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Surface hidden problems before expanding; never allow raw accumulation to masquerade as depth.
2. Distinguish symptoms from root knots with explicit boundaries, confidence tags, and failure conditions.
3. Map stakeholders or schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define tests of solution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 198 - Problem Cartography Layer: Update and Versioning Rule

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is update and versioning rule, which means the model must distinguish symptoms from root knots, map stakeholders or schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Distinguish symptoms from root knots before expanding; never allow raw accumulation to masquerade as depth.
2. Map stakeholders or schools with explicit boundaries, confidence tags, and failure conditions.
3. Define tests of solution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Surface hidden problems through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 199 - Problem Cartography Layer: Novelty and Hypothesis Gate

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is novelty and hypothesis gate, which means the model must map stakeholders or schools, define tests of solution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map stakeholders or schools before expanding; never allow raw accumulation to masquerade as depth.
2. Define tests of solution with explicit boundaries, confidence tags, and failure conditions.
3. Surface hidden problems by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Distinguish symptoms from root knots through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 200 - Problem Cartography Layer: Final Prompt Clause

This page belongs to Part 10: Problem Cartography Layer. Its task is to maps the visible and hidden problems that make the field difficult, generative, and worth engineering. The local module is final prompt clause, which means the model must define tests of solution, surface hidden problems, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define tests of solution before expanding; never allow raw accumulation to masquerade as depth.
2. Surface hidden problems with explicit boundaries, confidence tags, and failure conditions.
3. Distinguish symptoms from root knots by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map stakeholders or schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 201 - Method and Protocol Layer: Definition of the Operating Unit

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is definition of the operating unit, which means the model must write repeatable steps, define input and output, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write repeatable steps before expanding; never allow raw accumulation to masquerade as depth.
2. Define input and output with explicit boundaries, confidence tags, and failure conditions.
3. Name failure conditions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make method portable through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 202 - Method and Protocol Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is scope boundary and non-scope guard, which means the model must define input and output, name failure conditions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define input and output before expanding; never allow raw accumulation to masquerade as depth.
2. Name failure conditions with explicit boundaries, confidence tags, and failure conditions.
3. Make method portable by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write repeatable steps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 203 - Method and Protocol Layer: Kernel Extraction Move

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is kernel extraction move, which means the model must name failure conditions, make method portable, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name failure conditions before expanding; never allow raw accumulation to masquerade as depth.
2. Make method portable with explicit boundaries, confidence tags, and failure conditions.
3. Write repeatable steps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define input and output through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 204 - Method and Protocol Layer: Difference-Making Test

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is difference-making test, which means the model must make method portable, write repeatable steps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make method portable before expanding; never allow raw accumulation to masquerade as depth.
2. Write repeatable steps with explicit boundaries, confidence tags, and failure conditions.
3. Define input and output by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name failure conditions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 205 - Method and Protocol Layer: Confidence and Evidence Tagging

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is confidence and evidence tagging, which means the model must write repeatable steps, define input and output, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write repeatable steps before expanding; never allow raw accumulation to masquerade as depth.
2. Define input and output with explicit boundaries, confidence tags, and failure conditions.
3. Name failure conditions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make method portable through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 206 - Method and Protocol Layer: Concept Boundary Split

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is concept boundary split, which means the model must define input and output, name failure conditions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define input and output before expanding; never allow raw accumulation to masquerade as depth.
2. Name failure conditions with explicit boundaries, confidence tags, and failure conditions.
3. Make method portable by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write repeatable steps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 207 - Method and Protocol Layer: Problem Deepening Method

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is problem deepening method, which means the model must name failure conditions, make method portable, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name failure conditions before expanding; never allow raw accumulation to masquerade as depth.
2. Make method portable with explicit boundaries, confidence tags, and failure conditions.
3. Write repeatable steps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define input and output through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 208 - Method and Protocol Layer: Skill Conversion Move

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is skill conversion move, which means the model must make method portable, write repeatable steps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make method portable before expanding; never allow raw accumulation to masquerade as depth.
2. Write repeatable steps with explicit boundaries, confidence tags, and failure conditions.
3. Define input and output by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name failure conditions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 209 - Method and Protocol Layer: Loop Wiring Move

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is loop wiring move, which means the model must write repeatable steps, define input and output, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write repeatable steps before expanding; never allow raw accumulation to masquerade as depth.
2. Define input and output with explicit boundaries, confidence tags, and failure conditions.
3. Name failure conditions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make method portable through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 210 - Method and Protocol Layer: Tool Conversion Move

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is tool conversion move, which means the model must define input and output, name failure conditions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define input and output before expanding; never allow raw accumulation to masquerade as depth.
2. Name failure conditions with explicit boundaries, confidence tags, and failure conditions.
3. Make method portable by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write repeatable steps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 211 - Method and Protocol Layer: Retrieval and Metadata Move

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is retrieval and metadata move, which means the model must name failure conditions, make method portable, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name failure conditions before expanding; never allow raw accumulation to masquerade as depth.
2. Make method portable with explicit boundaries, confidence tags, and failure conditions.
3. Write repeatable steps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define input and output through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 212 - Method and Protocol Layer: Agent Role Assignment

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is agent role assignment, which means the model must make method portable, write repeatable steps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make method portable before expanding; never allow raw accumulation to masquerade as depth.
2. Write repeatable steps with explicit boundaries, confidence tags, and failure conditions.
3. Define input and output by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name failure conditions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 213 - Method and Protocol Layer: Court of Destruction Pass

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is court of destruction pass, which means the model must write repeatable steps, define input and output, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write repeatable steps before expanding; never allow raw accumulation to masquerade as depth.
2. Define input and output with explicit boundaries, confidence tags, and failure conditions.
3. Name failure conditions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make method portable through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 214 - Method and Protocol Layer: Repair and Reconstruction Pass

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is repair and reconstruction pass, which means the model must define input and output, name failure conditions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define input and output before expanding; never allow raw accumulation to masquerade as depth.
2. Name failure conditions with explicit boundaries, confidence tags, and failure conditions.
3. Make method portable by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write repeatable steps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 215 - Method and Protocol Layer: Training and Mastery Drill

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is training and mastery drill, which means the model must name failure conditions, make method portable, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name failure conditions before expanding; never allow raw accumulation to masquerade as depth.
2. Make method portable with explicit boundaries, confidence tags, and failure conditions.
3. Write repeatable steps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define input and output through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 216 - Method and Protocol Layer: Output Interface Contract

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is output interface contract, which means the model must make method portable, write repeatable steps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make method portable before expanding; never allow raw accumulation to masquerade as depth.
2. Write repeatable steps with explicit boundaries, confidence tags, and failure conditions.
3. Define input and output by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name failure conditions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 217 - Method and Protocol Layer: Failure Mode Catalogue

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is failure mode catalogue, which means the model must write repeatable steps, define input and output, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write repeatable steps before expanding; never allow raw accumulation to masquerade as depth.
2. Define input and output with explicit boundaries, confidence tags, and failure conditions.
3. Name failure conditions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make method portable through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 218 - Method and Protocol Layer: Update and Versioning Rule

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is update and versioning rule, which means the model must define input and output, name failure conditions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define input and output before expanding; never allow raw accumulation to masquerade as depth.
2. Name failure conditions with explicit boundaries, confidence tags, and failure conditions.
3. Make method portable by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write repeatable steps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 219 - Method and Protocol Layer: Novelty and Hypothesis Gate

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is novelty and hypothesis gate, which means the model must name failure conditions, make method portable, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Name failure conditions before expanding; never allow raw accumulation to masquerade as depth.
2. Make method portable with explicit boundaries, confidence tags, and failure conditions.
3. Write repeatable steps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define input and output through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 220 - Method and Protocol Layer: Final Prompt Clause

This page belongs to Part 11: Method and Protocol Layer. Its task is to builds repeatable reasoning procedures that can be used by humans, models, and agents. The local module is final prompt clause, which means the model must make method portable, write repeatable steps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make method portable before expanding; never allow raw accumulation to masquerade as depth.
2. Write repeatable steps with explicit boundaries, confidence tags, and failure conditions.
3. Define input and output by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Name failure conditions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 221 - HyperSkills Architecture: Definition of the Operating Unit

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is definition of the operating unit, which means the model must extract trainable skills, define mastery signs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract trainable skills before expanding; never allow raw accumulation to masquerade as depth.
2. Define mastery signs with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Connect skill to loops through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 222 - HyperSkills Architecture: Scope Boundary and Non-Scope Guard

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is scope boundary and non-scope guard, which means the model must define mastery signs, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define mastery signs before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Connect skill to loops by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract trainable skills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 223 - HyperSkills Architecture: Kernel Extraction Move

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is kernel extraction move, which means the model must write drills, connect skill to loops, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Connect skill to loops with explicit boundaries, confidence tags, and failure conditions.
3. Extract trainable skills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define mastery signs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 224 - HyperSkills Architecture: Difference-Making Test

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is difference-making test, which means the model must connect skill to loops, extract trainable skills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Connect skill to loops before expanding; never allow raw accumulation to masquerade as depth.
2. Extract trainable skills with explicit boundaries, confidence tags, and failure conditions.
3. Define mastery signs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 225 - HyperSkills Architecture: Confidence and Evidence Tagging

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is confidence and evidence tagging, which means the model must extract trainable skills, define mastery signs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract trainable skills before expanding; never allow raw accumulation to masquerade as depth.
2. Define mastery signs with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Connect skill to loops through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 226 - HyperSkills Architecture: Concept Boundary Split

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is concept boundary split, which means the model must define mastery signs, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define mastery signs before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Connect skill to loops by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract trainable skills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 227 - HyperSkills Architecture: Problem Deepening Method

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is problem deepening method, which means the model must write drills, connect skill to loops, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Connect skill to loops with explicit boundaries, confidence tags, and failure conditions.
3. Extract trainable skills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define mastery signs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 228 - HyperSkills Architecture: Skill Conversion Move

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is skill conversion move, which means the model must connect skill to loops, extract trainable skills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Connect skill to loops before expanding; never allow raw accumulation to masquerade as depth.
2. Extract trainable skills with explicit boundaries, confidence tags, and failure conditions.
3. Define mastery signs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 229 - HyperSkills Architecture: Loop Wiring Move

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is loop wiring move, which means the model must extract trainable skills, define mastery signs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract trainable skills before expanding; never allow raw accumulation to masquerade as depth.
2. Define mastery signs with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Connect skill to loops through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 230 - HyperSkills Architecture: Tool Conversion Move

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is tool conversion move, which means the model must define mastery signs, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define mastery signs before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Connect skill to loops by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract trainable skills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 231 - HyperSkills Architecture: Retrieval and Metadata Move

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is retrieval and metadata move, which means the model must write drills, connect skill to loops, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Connect skill to loops with explicit boundaries, confidence tags, and failure conditions.
3. Extract trainable skills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define mastery signs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 232 - HyperSkills Architecture: Agent Role Assignment

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is agent role assignment, which means the model must connect skill to loops, extract trainable skills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Connect skill to loops before expanding; never allow raw accumulation to masquerade as depth.
2. Extract trainable skills with explicit boundaries, confidence tags, and failure conditions.
3. Define mastery signs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 233 - HyperSkills Architecture: Court of Destruction Pass

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is court of destruction pass, which means the model must extract trainable skills, define mastery signs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract trainable skills before expanding; never allow raw accumulation to masquerade as depth.
2. Define mastery signs with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Connect skill to loops through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 234 - HyperSkills Architecture: Repair and Reconstruction Pass

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is repair and reconstruction pass, which means the model must define mastery signs, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define mastery signs before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Connect skill to loops by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract trainable skills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 235 - HyperSkills Architecture: Training and Mastery Drill

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is training and mastery drill, which means the model must write drills, connect skill to loops, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Connect skill to loops with explicit boundaries, confidence tags, and failure conditions.
3. Extract trainable skills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define mastery signs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 236 - HyperSkills Architecture: Output Interface Contract

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is output interface contract, which means the model must connect skill to loops, extract trainable skills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Connect skill to loops before expanding; never allow raw accumulation to masquerade as depth.
2. Extract trainable skills with explicit boundaries, confidence tags, and failure conditions.
3. Define mastery signs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 237 - HyperSkills Architecture: Failure Mode Catalogue

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is failure mode catalogue, which means the model must extract trainable skills, define mastery signs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Extract trainable skills before expanding; never allow raw accumulation to masquerade as depth.
2. Define mastery signs with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Connect skill to loops through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 238 - HyperSkills Architecture: Update and Versioning Rule

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is update and versioning rule, which means the model must define mastery signs, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define mastery signs before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Connect skill to loops by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Extract trainable skills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 239 - HyperSkills Architecture: Novelty and Hypothesis Gate

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is novelty and hypothesis gate, which means the model must write drills, connect skill to loops, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Connect skill to loops with explicit boundaries, confidence tags, and failure conditions.
3. Extract trainable skills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define mastery signs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 240 - HyperSkills Architecture: Final Prompt Clause

This page belongs to Part 12: HyperSkills Architecture. Its task is to extracts trainable super-skills from the field and turns understanding into reliable capability. The local module is final prompt clause, which means the model must connect skill to loops, extract trainable skills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Connect skill to loops before expanding; never allow raw accumulation to masquerade as depth.
2. Extract trainable skills with explicit boundaries, confidence tags, and failure conditions.
3. Define mastery signs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 241 - HyperLoops Architecture: Definition of the Operating Unit

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is definition of the operating unit, which means the model must define recurring cycle, state trigger and cadence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define recurring cycle before expanding; never allow raw accumulation to masquerade as depth.
2. State trigger and cadence with explicit boundaries, confidence tags, and failure conditions.
3. Measure improvement by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Store repair deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 242 - HyperLoops Architecture: Scope Boundary and Non-Scope Guard

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is scope boundary and non-scope guard, which means the model must state trigger and cadence, measure improvement, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State trigger and cadence before expanding; never allow raw accumulation to masquerade as depth.
2. Measure improvement with explicit boundaries, confidence tags, and failure conditions.
3. Store repair deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define recurring cycle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 243 - HyperLoops Architecture: Kernel Extraction Move

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is kernel extraction move, which means the model must measure improvement, store repair deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure improvement before expanding; never allow raw accumulation to masquerade as depth.
2. Store repair deltas with explicit boundaries, confidence tags, and failure conditions.
3. Define recurring cycle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State trigger and cadence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 244 - HyperLoops Architecture: Difference-Making Test

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is difference-making test, which means the model must store repair deltas, define recurring cycle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Store repair deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Define recurring cycle with explicit boundaries, confidence tags, and failure conditions.
3. State trigger and cadence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure improvement through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 245 - HyperLoops Architecture: Confidence and Evidence Tagging

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is confidence and evidence tagging, which means the model must define recurring cycle, state trigger and cadence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define recurring cycle before expanding; never allow raw accumulation to masquerade as depth.
2. State trigger and cadence with explicit boundaries, confidence tags, and failure conditions.
3. Measure improvement by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Store repair deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 246 - HyperLoops Architecture: Concept Boundary Split

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is concept boundary split, which means the model must state trigger and cadence, measure improvement, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State trigger and cadence before expanding; never allow raw accumulation to masquerade as depth.
2. Measure improvement with explicit boundaries, confidence tags, and failure conditions.
3. Store repair deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define recurring cycle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 247 - HyperLoops Architecture: Problem Deepening Method

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is problem deepening method, which means the model must measure improvement, store repair deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure improvement before expanding; never allow raw accumulation to masquerade as depth.
2. Store repair deltas with explicit boundaries, confidence tags, and failure conditions.
3. Define recurring cycle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State trigger and cadence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 248 - HyperLoops Architecture: Skill Conversion Move

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is skill conversion move, which means the model must store repair deltas, define recurring cycle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Store repair deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Define recurring cycle with explicit boundaries, confidence tags, and failure conditions.
3. State trigger and cadence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure improvement through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 249 - HyperLoops Architecture: Loop Wiring Move

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is loop wiring move, which means the model must define recurring cycle, state trigger and cadence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define recurring cycle before expanding; never allow raw accumulation to masquerade as depth.
2. State trigger and cadence with explicit boundaries, confidence tags, and failure conditions.
3. Measure improvement by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Store repair deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 250 - HyperLoops Architecture: Tool Conversion Move

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is tool conversion move, which means the model must state trigger and cadence, measure improvement, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State trigger and cadence before expanding; never allow raw accumulation to masquerade as depth.
2. Measure improvement with explicit boundaries, confidence tags, and failure conditions.
3. Store repair deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define recurring cycle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 251 - HyperLoops Architecture: Retrieval and Metadata Move

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is retrieval and metadata move, which means the model must measure improvement, store repair deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure improvement before expanding; never allow raw accumulation to masquerade as depth.
2. Store repair deltas with explicit boundaries, confidence tags, and failure conditions.
3. Define recurring cycle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State trigger and cadence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 252 - HyperLoops Architecture: Agent Role Assignment

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is agent role assignment, which means the model must store repair deltas, define recurring cycle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Store repair deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Define recurring cycle with explicit boundaries, confidence tags, and failure conditions.
3. State trigger and cadence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure improvement through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 253 - HyperLoops Architecture: Court of Destruction Pass

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is court of destruction pass, which means the model must define recurring cycle, state trigger and cadence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define recurring cycle before expanding; never allow raw accumulation to masquerade as depth.
2. State trigger and cadence with explicit boundaries, confidence tags, and failure conditions.
3. Measure improvement by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Store repair deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 254 - HyperLoops Architecture: Repair and Reconstruction Pass

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is repair and reconstruction pass, which means the model must state trigger and cadence, measure improvement, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State trigger and cadence before expanding; never allow raw accumulation to masquerade as depth.
2. Measure improvement with explicit boundaries, confidence tags, and failure conditions.
3. Store repair deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define recurring cycle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 255 - HyperLoops Architecture: Training and Mastery Drill

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is training and mastery drill, which means the model must measure improvement, store repair deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure improvement before expanding; never allow raw accumulation to masquerade as depth.
2. Store repair deltas with explicit boundaries, confidence tags, and failure conditions.
3. Define recurring cycle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State trigger and cadence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 256 - HyperLoops Architecture: Output Interface Contract

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is output interface contract, which means the model must store repair deltas, define recurring cycle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Store repair deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Define recurring cycle with explicit boundaries, confidence tags, and failure conditions.
3. State trigger and cadence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure improvement through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 257 - HyperLoops Architecture: Failure Mode Catalogue

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is failure mode catalogue, which means the model must define recurring cycle, state trigger and cadence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define recurring cycle before expanding; never allow raw accumulation to masquerade as depth.
2. State trigger and cadence with explicit boundaries, confidence tags, and failure conditions.
3. Measure improvement by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Store repair deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 258 - HyperLoops Architecture: Update and Versioning Rule

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is update and versioning rule, which means the model must state trigger and cadence, measure improvement, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State trigger and cadence before expanding; never allow raw accumulation to masquerade as depth.
2. Measure improvement with explicit boundaries, confidence tags, and failure conditions.
3. Store repair deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define recurring cycle through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 259 - HyperLoops Architecture: Novelty and Hypothesis Gate

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is novelty and hypothesis gate, which means the model must measure improvement, store repair deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure improvement before expanding; never allow raw accumulation to masquerade as depth.
2. Store repair deltas with explicit boundaries, confidence tags, and failure conditions.
3. Define recurring cycle by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State trigger and cadence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 260 - HyperLoops Architecture: Final Prompt Clause

This page belongs to Part 13: HyperLoops Architecture. Its task is to builds learning, critique, repair, update, production, and re-entry loops that keep the base alive. The local module is final prompt clause, which means the model must store repair deltas, define recurring cycle, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Store repair deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Define recurring cycle with explicit boundaries, confidence tags, and failure conditions.
3. State trigger and cadence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure improvement through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 261 - Tool Forge Layer: Definition of the Operating Unit

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is definition of the operating unit, which means the model must build matrices, write checklists, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build matrices before expanding; never allow raw accumulation to masquerade as depth.
2. Write checklists with explicit boundaries, confidence tags, and failure conditions.
3. Make algorithms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test usability through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 262 - Tool Forge Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is scope boundary and non-scope guard, which means the model must write checklists, make algorithms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write checklists before expanding; never allow raw accumulation to masquerade as depth.
2. Make algorithms with explicit boundaries, confidence tags, and failure conditions.
3. Test usability by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build matrices through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 263 - Tool Forge Layer: Kernel Extraction Move

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is kernel extraction move, which means the model must make algorithms, test usability, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make algorithms before expanding; never allow raw accumulation to masquerade as depth.
2. Test usability with explicit boundaries, confidence tags, and failure conditions.
3. Build matrices by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write checklists through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 264 - Tool Forge Layer: Difference-Making Test

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is difference-making test, which means the model must test usability, build matrices, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test usability before expanding; never allow raw accumulation to masquerade as depth.
2. Build matrices with explicit boundaries, confidence tags, and failure conditions.
3. Write checklists by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make algorithms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 265 - Tool Forge Layer: Confidence and Evidence Tagging

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is confidence and evidence tagging, which means the model must build matrices, write checklists, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build matrices before expanding; never allow raw accumulation to masquerade as depth.
2. Write checklists with explicit boundaries, confidence tags, and failure conditions.
3. Make algorithms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test usability through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 266 - Tool Forge Layer: Concept Boundary Split

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is concept boundary split, which means the model must write checklists, make algorithms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write checklists before expanding; never allow raw accumulation to masquerade as depth.
2. Make algorithms with explicit boundaries, confidence tags, and failure conditions.
3. Test usability by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build matrices through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 267 - Tool Forge Layer: Problem Deepening Method

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is problem deepening method, which means the model must make algorithms, test usability, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make algorithms before expanding; never allow raw accumulation to masquerade as depth.
2. Test usability with explicit boundaries, confidence tags, and failure conditions.
3. Build matrices by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write checklists through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 268 - Tool Forge Layer: Skill Conversion Move

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is skill conversion move, which means the model must test usability, build matrices, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test usability before expanding; never allow raw accumulation to masquerade as depth.
2. Build matrices with explicit boundaries, confidence tags, and failure conditions.
3. Write checklists by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make algorithms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 269 - Tool Forge Layer: Loop Wiring Move

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is loop wiring move, which means the model must build matrices, write checklists, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build matrices before expanding; never allow raw accumulation to masquerade as depth.
2. Write checklists with explicit boundaries, confidence tags, and failure conditions.
3. Make algorithms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test usability through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 270 - Tool Forge Layer: Tool Conversion Move

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is tool conversion move, which means the model must write checklists, make algorithms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write checklists before expanding; never allow raw accumulation to masquerade as depth.
2. Make algorithms with explicit boundaries, confidence tags, and failure conditions.
3. Test usability by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build matrices through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 271 - Tool Forge Layer: Retrieval and Metadata Move

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is retrieval and metadata move, which means the model must make algorithms, test usability, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make algorithms before expanding; never allow raw accumulation to masquerade as depth.
2. Test usability with explicit boundaries, confidence tags, and failure conditions.
3. Build matrices by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write checklists through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 272 - Tool Forge Layer: Agent Role Assignment

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is agent role assignment, which means the model must test usability, build matrices, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test usability before expanding; never allow raw accumulation to masquerade as depth.
2. Build matrices with explicit boundaries, confidence tags, and failure conditions.
3. Write checklists by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make algorithms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 273 - Tool Forge Layer: Court of Destruction Pass

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is court of destruction pass, which means the model must build matrices, write checklists, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build matrices before expanding; never allow raw accumulation to masquerade as depth.
2. Write checklists with explicit boundaries, confidence tags, and failure conditions.
3. Make algorithms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test usability through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 274 - Tool Forge Layer: Repair and Reconstruction Pass

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is repair and reconstruction pass, which means the model must write checklists, make algorithms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write checklists before expanding; never allow raw accumulation to masquerade as depth.
2. Make algorithms with explicit boundaries, confidence tags, and failure conditions.
3. Test usability by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build matrices through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 275 - Tool Forge Layer: Training and Mastery Drill

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is training and mastery drill, which means the model must make algorithms, test usability, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make algorithms before expanding; never allow raw accumulation to masquerade as depth.
2. Test usability with explicit boundaries, confidence tags, and failure conditions.
3. Build matrices by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write checklists through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 276 - Tool Forge Layer: Output Interface Contract

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is output interface contract, which means the model must test usability, build matrices, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test usability before expanding; never allow raw accumulation to masquerade as depth.
2. Build matrices with explicit boundaries, confidence tags, and failure conditions.
3. Write checklists by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make algorithms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 277 - Tool Forge Layer: Failure Mode Catalogue

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is failure mode catalogue, which means the model must build matrices, write checklists, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build matrices before expanding; never allow raw accumulation to masquerade as depth.
2. Write checklists with explicit boundaries, confidence tags, and failure conditions.
3. Make algorithms by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test usability through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 278 - Tool Forge Layer: Update and Versioning Rule

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is update and versioning rule, which means the model must write checklists, make algorithms, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write checklists before expanding; never allow raw accumulation to masquerade as depth.
2. Make algorithms with explicit boundaries, confidence tags, and failure conditions.
3. Test usability by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build matrices through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 279 - Tool Forge Layer: Novelty and Hypothesis Gate

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is novelty and hypothesis gate, which means the model must make algorithms, test usability, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Make algorithms before expanding; never allow raw accumulation to masquerade as depth.
2. Test usability with explicit boundaries, confidence tags, and failure conditions.
3. Build matrices by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write checklists through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 280 - Tool Forge Layer: Final Prompt Clause

This page belongs to Part 14: Tool Forge Layer. Its task is to converts concepts and methods into matrices, checklists, algorithms, evaluators, and workflows. The local module is final prompt clause, which means the model must test usability, build matrices, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test usability before expanding; never allow raw accumulation to masquerade as depth.
2. Build matrices with explicit boundaries, confidence tags, and failure conditions.
3. Write checklists by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Make algorithms through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 281 - Prompt Operating System: Definition of the Operating Unit

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is definition of the operating unit, which means the model must write system clauses, create task contracts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write system clauses before expanding; never allow raw accumulation to masquerade as depth.
2. Create task contracts with explicit boundaries, confidence tags, and failure conditions.
3. Assemble prompt modules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Prevent ambiguous execution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 282 - Prompt Operating System: Scope Boundary and Non-Scope Guard

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is scope boundary and non-scope guard, which means the model must create task contracts, assemble prompt modules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create task contracts before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble prompt modules with explicit boundaries, confidence tags, and failure conditions.
3. Prevent ambiguous execution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write system clauses through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 283 - Prompt Operating System: Kernel Extraction Move

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is kernel extraction move, which means the model must assemble prompt modules, prevent ambiguous execution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble prompt modules before expanding; never allow raw accumulation to masquerade as depth.
2. Prevent ambiguous execution with explicit boundaries, confidence tags, and failure conditions.
3. Write system clauses by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create task contracts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 284 - Prompt Operating System: Difference-Making Test

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is difference-making test, which means the model must prevent ambiguous execution, write system clauses, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Prevent ambiguous execution before expanding; never allow raw accumulation to masquerade as depth.
2. Write system clauses with explicit boundaries, confidence tags, and failure conditions.
3. Create task contracts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble prompt modules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 285 - Prompt Operating System: Confidence and Evidence Tagging

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is confidence and evidence tagging, which means the model must write system clauses, create task contracts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write system clauses before expanding; never allow raw accumulation to masquerade as depth.
2. Create task contracts with explicit boundaries, confidence tags, and failure conditions.
3. Assemble prompt modules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Prevent ambiguous execution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 286 - Prompt Operating System: Concept Boundary Split

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is concept boundary split, which means the model must create task contracts, assemble prompt modules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create task contracts before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble prompt modules with explicit boundaries, confidence tags, and failure conditions.
3. Prevent ambiguous execution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write system clauses through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 287 - Prompt Operating System: Problem Deepening Method

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is problem deepening method, which means the model must assemble prompt modules, prevent ambiguous execution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble prompt modules before expanding; never allow raw accumulation to masquerade as depth.
2. Prevent ambiguous execution with explicit boundaries, confidence tags, and failure conditions.
3. Write system clauses by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create task contracts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 288 - Prompt Operating System: Skill Conversion Move

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is skill conversion move, which means the model must prevent ambiguous execution, write system clauses, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Prevent ambiguous execution before expanding; never allow raw accumulation to masquerade as depth.
2. Write system clauses with explicit boundaries, confidence tags, and failure conditions.
3. Create task contracts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble prompt modules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 289 - Prompt Operating System: Loop Wiring Move

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is loop wiring move, which means the model must write system clauses, create task contracts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must index the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write system clauses before expanding; never allow raw accumulation to masquerade as depth.
2. Create task contracts with explicit boundaries, confidence tags, and failure conditions.
3. Assemble prompt modules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Prevent ambiguous execution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 290 - Prompt Operating System: Tool Conversion Move

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is tool conversion move, which means the model must create task contracts, assemble prompt modules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must train the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create task contracts before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble prompt modules with explicit boundaries, confidence tags, and failure conditions.
3. Prevent ambiguous execution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write system clauses through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 291 - Prompt Operating System: Retrieval and Metadata Move

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is retrieval and metadata move, which means the model must assemble prompt modules, prevent ambiguous execution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must extract the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble prompt modules before expanding; never allow raw accumulation to masquerade as depth.
2. Prevent ambiguous execution with explicit boundaries, confidence tags, and failure conditions.
3. Write system clauses by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create task contracts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 292 - Prompt Operating System: Agent Role Assignment

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is agent role assignment, which means the model must prevent ambiguous execution, write system clauses, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must compress the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Prevent ambiguous execution before expanding; never allow raw accumulation to masquerade as depth.
2. Write system clauses with explicit boundaries, confidence tags, and failure conditions.
3. Create task contracts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble prompt modules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 293 - Prompt Operating System: Court of Destruction Pass

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is court of destruction pass, which means the model must write system clauses, create task contracts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write system clauses before expanding; never allow raw accumulation to masquerade as depth.
2. Create task contracts with explicit boundaries, confidence tags, and failure conditions.
3. Assemble prompt modules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Prevent ambiguous execution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 294 - Prompt Operating System: Repair and Reconstruction Pass

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is repair and reconstruction pass, which means the model must create task contracts, assemble prompt modules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create task contracts before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble prompt modules with explicit boundaries, confidence tags, and failure conditions.
3. Prevent ambiguous execution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write system clauses through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 295 - Prompt Operating System: Training and Mastery Drill

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is training and mastery drill, which means the model must assemble prompt modules, prevent ambiguous execution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble prompt modules before expanding; never allow raw accumulation to masquerade as depth.
2. Prevent ambiguous execution with explicit boundaries, confidence tags, and failure conditions.
3. Write system clauses by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create task contracts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 296 - Prompt Operating System: Output Interface Contract

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is output interface contract, which means the model must prevent ambiguous execution, write system clauses, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Prevent ambiguous execution before expanding; never allow raw accumulation to masquerade as depth.
2. Write system clauses with explicit boundaries, confidence tags, and failure conditions.
3. Create task contracts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble prompt modules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 297 - Prompt Operating System: Failure Mode Catalogue

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is failure mode catalogue, which means the model must write system clauses, create task contracts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write system clauses before expanding; never allow raw accumulation to masquerade as depth.
2. Create task contracts with explicit boundaries, confidence tags, and failure conditions.
3. Assemble prompt modules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Prevent ambiguous execution through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 298 - Prompt Operating System: Update and Versioning Rule

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is update and versioning rule, which means the model must create task contracts, assemble prompt modules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create task contracts before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble prompt modules with explicit boundaries, confidence tags, and failure conditions.
3. Prevent ambiguous execution by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write system clauses through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 299 - Prompt Operating System: Novelty and Hypothesis Gate

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is novelty and hypothesis gate, which means the model must assemble prompt modules, prevent ambiguous execution, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble prompt modules before expanding; never allow raw accumulation to masquerade as depth.
2. Prevent ambiguous execution with explicit boundaries, confidence tags, and failure conditions.
3. Write system clauses by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create task contracts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 300 - Prompt Operating System: Final Prompt Clause

This page belongs to Part 15: Prompt Operating System. Its task is to turns the base into reusable prompt modules, system cards, task contracts, and execution clauses. The local module is final prompt clause, which means the model must prevent ambiguous execution, write system clauses, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Prevent ambiguous execution before expanding; never allow raw accumulation to masquerade as depth.
2. Write system clauses with explicit boundaries, confidence tags, and failure conditions.
3. Create task contracts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble prompt modules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 301 - RAG and Chunking Design: Definition of the Operating Unit

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is definition of the operating unit, which means the model must chunk by unit type, tag metadata, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Chunk by unit type before expanding; never allow raw accumulation to masquerade as depth.
2. Tag metadata with explicit boundaries, confidence tags, and failure conditions.
3. Preserve evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route query intent through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 302 - RAG and Chunking Design: Scope Boundary and Non-Scope Guard

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is scope boundary and non-scope guard, which means the model must tag metadata, preserve evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag metadata before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve evidence with explicit boundaries, confidence tags, and failure conditions.
3. Route query intent by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Chunk by unit type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 303 - RAG and Chunking Design: Kernel Extraction Move

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is kernel extraction move, which means the model must preserve evidence, route query intent, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Route query intent with explicit boundaries, confidence tags, and failure conditions.
3. Chunk by unit type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag metadata through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 304 - RAG and Chunking Design: Difference-Making Test

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is difference-making test, which means the model must route query intent, chunk by unit type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route query intent before expanding; never allow raw accumulation to masquerade as depth.
2. Chunk by unit type with explicit boundaries, confidence tags, and failure conditions.
3. Tag metadata by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 305 - RAG and Chunking Design: Confidence and Evidence Tagging

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is confidence and evidence tagging, which means the model must chunk by unit type, tag metadata, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Chunk by unit type before expanding; never allow raw accumulation to masquerade as depth.
2. Tag metadata with explicit boundaries, confidence tags, and failure conditions.
3. Preserve evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route query intent through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 306 - RAG and Chunking Design: Concept Boundary Split

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is concept boundary split, which means the model must tag metadata, preserve evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag metadata before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve evidence with explicit boundaries, confidence tags, and failure conditions.
3. Route query intent by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Chunk by unit type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 307 - RAG and Chunking Design: Problem Deepening Method

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is problem deepening method, which means the model must preserve evidence, route query intent, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Route query intent with explicit boundaries, confidence tags, and failure conditions.
3. Chunk by unit type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag metadata through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 308 - RAG and Chunking Design: Skill Conversion Move

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is skill conversion move, which means the model must route query intent, chunk by unit type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route query intent before expanding; never allow raw accumulation to masquerade as depth.
2. Chunk by unit type with explicit boundaries, confidence tags, and failure conditions.
3. Tag metadata by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 309 - RAG and Chunking Design: Loop Wiring Move

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is loop wiring move, which means the model must chunk by unit type, tag metadata, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must train the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Chunk by unit type before expanding; never allow raw accumulation to masquerade as depth.
2. Tag metadata with explicit boundaries, confidence tags, and failure conditions.
3. Preserve evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route query intent through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 310 - RAG and Chunking Design: Tool Conversion Move

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is tool conversion move, which means the model must tag metadata, preserve evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must extract the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag metadata before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve evidence with explicit boundaries, confidence tags, and failure conditions.
3. Route query intent by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Chunk by unit type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 311 - RAG and Chunking Design: Retrieval and Metadata Move

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is retrieval and metadata move, which means the model must preserve evidence, route query intent, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must compress the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Route query intent with explicit boundaries, confidence tags, and failure conditions.
3. Chunk by unit type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag metadata through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 312 - RAG and Chunking Design: Agent Role Assignment

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is agent role assignment, which means the model must route query intent, chunk by unit type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must map the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route query intent before expanding; never allow raw accumulation to masquerade as depth.
2. Chunk by unit type with explicit boundaries, confidence tags, and failure conditions.
3. Tag metadata by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 313 - RAG and Chunking Design: Court of Destruction Pass

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is court of destruction pass, which means the model must chunk by unit type, tag metadata, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Chunk by unit type before expanding; never allow raw accumulation to masquerade as depth.
2. Tag metadata with explicit boundaries, confidence tags, and failure conditions.
3. Preserve evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route query intent through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 314 - RAG and Chunking Design: Repair and Reconstruction Pass

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is repair and reconstruction pass, which means the model must tag metadata, preserve evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag metadata before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve evidence with explicit boundaries, confidence tags, and failure conditions.
3. Route query intent by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Chunk by unit type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 315 - RAG and Chunking Design: Training and Mastery Drill

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is training and mastery drill, which means the model must preserve evidence, route query intent, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Route query intent with explicit boundaries, confidence tags, and failure conditions.
3. Chunk by unit type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag metadata through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 316 - RAG and Chunking Design: Output Interface Contract

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is output interface contract, which means the model must route query intent, chunk by unit type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route query intent before expanding; never allow raw accumulation to masquerade as depth.
2. Chunk by unit type with explicit boundaries, confidence tags, and failure conditions.
3. Tag metadata by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 317 - RAG and Chunking Design: Failure Mode Catalogue

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is failure mode catalogue, which means the model must chunk by unit type, tag metadata, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Chunk by unit type before expanding; never allow raw accumulation to masquerade as depth.
2. Tag metadata with explicit boundaries, confidence tags, and failure conditions.
3. Preserve evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Route query intent through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 318 - RAG and Chunking Design: Update and Versioning Rule

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is update and versioning rule, which means the model must tag metadata, preserve evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Tag metadata before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve evidence with explicit boundaries, confidence tags, and failure conditions.
3. Route query intent by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Chunk by unit type through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 319 - RAG and Chunking Design: Novelty and Hypothesis Gate

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is novelty and hypothesis gate, which means the model must preserve evidence, route query intent, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Route query intent with explicit boundaries, confidence tags, and failure conditions.
3. Chunk by unit type by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Tag metadata through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 320 - RAG and Chunking Design: Final Prompt Clause

This page belongs to Part 16: RAG and Chunking Design. Its task is to prepares the base for retrieval, chunking, metadata, confidence filtering, and query routing. The local module is final prompt clause, which means the model must route query intent, chunk by unit type, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Route query intent before expanding; never allow raw accumulation to masquerade as depth.
2. Chunk by unit type with explicit boundaries, confidence tags, and failure conditions.
3. Tag metadata by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 321 - GraphRAG and Relation Design: Definition of the Operating Unit

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is definition of the operating unit, which means the model must create nodes, type edges, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create nodes before expanding; never allow raw accumulation to masquerade as depth.
2. Type edges with explicit boundaries, confidence tags, and failure conditions.
3. Mark contradictions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Trace evidence paths through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 322 - GraphRAG and Relation Design: Scope Boundary and Non-Scope Guard

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is scope boundary and non-scope guard, which means the model must type edges, mark contradictions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Type edges before expanding; never allow raw accumulation to masquerade as depth.
2. Mark contradictions with explicit boundaries, confidence tags, and failure conditions.
3. Trace evidence paths by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create nodes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 323 - GraphRAG and Relation Design: Kernel Extraction Move

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is kernel extraction move, which means the model must mark contradictions, trace evidence paths, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark contradictions before expanding; never allow raw accumulation to masquerade as depth.
2. Trace evidence paths with explicit boundaries, confidence tags, and failure conditions.
3. Create nodes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Type edges through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 324 - GraphRAG and Relation Design: Difference-Making Test

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is difference-making test, which means the model must trace evidence paths, create nodes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Trace evidence paths before expanding; never allow raw accumulation to masquerade as depth.
2. Create nodes with explicit boundaries, confidence tags, and failure conditions.
3. Type edges by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark contradictions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 325 - GraphRAG and Relation Design: Confidence and Evidence Tagging

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is confidence and evidence tagging, which means the model must create nodes, type edges, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create nodes before expanding; never allow raw accumulation to masquerade as depth.
2. Type edges with explicit boundaries, confidence tags, and failure conditions.
3. Mark contradictions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Trace evidence paths through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 326 - GraphRAG and Relation Design: Concept Boundary Split

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is concept boundary split, which means the model must type edges, mark contradictions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Type edges before expanding; never allow raw accumulation to masquerade as depth.
2. Mark contradictions with explicit boundaries, confidence tags, and failure conditions.
3. Trace evidence paths by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create nodes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 327 - GraphRAG and Relation Design: Problem Deepening Method

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is problem deepening method, which means the model must mark contradictions, trace evidence paths, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark contradictions before expanding; never allow raw accumulation to masquerade as depth.
2. Trace evidence paths with explicit boundaries, confidence tags, and failure conditions.
3. Create nodes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Type edges through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 328 - GraphRAG and Relation Design: Skill Conversion Move

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is skill conversion move, which means the model must trace evidence paths, create nodes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Trace evidence paths before expanding; never allow raw accumulation to masquerade as depth.
2. Create nodes with explicit boundaries, confidence tags, and failure conditions.
3. Type edges by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark contradictions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 329 - GraphRAG and Relation Design: Loop Wiring Move

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is loop wiring move, which means the model must create nodes, type edges, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create nodes before expanding; never allow raw accumulation to masquerade as depth.
2. Type edges with explicit boundaries, confidence tags, and failure conditions.
3. Mark contradictions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Trace evidence paths through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 330 - GraphRAG and Relation Design: Tool Conversion Move

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is tool conversion move, which means the model must type edges, mark contradictions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Type edges before expanding; never allow raw accumulation to masquerade as depth.
2. Mark contradictions with explicit boundaries, confidence tags, and failure conditions.
3. Trace evidence paths by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create nodes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 331 - GraphRAG and Relation Design: Retrieval and Metadata Move

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is retrieval and metadata move, which means the model must mark contradictions, trace evidence paths, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark contradictions before expanding; never allow raw accumulation to masquerade as depth.
2. Trace evidence paths with explicit boundaries, confidence tags, and failure conditions.
3. Create nodes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Type edges through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 332 - GraphRAG and Relation Design: Agent Role Assignment

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is agent role assignment, which means the model must trace evidence paths, create nodes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Trace evidence paths before expanding; never allow raw accumulation to masquerade as depth.
2. Create nodes with explicit boundaries, confidence tags, and failure conditions.
3. Type edges by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark contradictions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 333 - GraphRAG and Relation Design: Court of Destruction Pass

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is court of destruction pass, which means the model must create nodes, type edges, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create nodes before expanding; never allow raw accumulation to masquerade as depth.
2. Type edges with explicit boundaries, confidence tags, and failure conditions.
3. Mark contradictions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Trace evidence paths through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 334 - GraphRAG and Relation Design: Repair and Reconstruction Pass

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is repair and reconstruction pass, which means the model must type edges, mark contradictions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Type edges before expanding; never allow raw accumulation to masquerade as depth.
2. Mark contradictions with explicit boundaries, confidence tags, and failure conditions.
3. Trace evidence paths by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create nodes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 335 - GraphRAG and Relation Design: Training and Mastery Drill

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is training and mastery drill, which means the model must mark contradictions, trace evidence paths, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark contradictions before expanding; never allow raw accumulation to masquerade as depth.
2. Trace evidence paths with explicit boundaries, confidence tags, and failure conditions.
3. Create nodes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Type edges through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 336 - GraphRAG and Relation Design: Output Interface Contract

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is output interface contract, which means the model must trace evidence paths, create nodes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Trace evidence paths before expanding; never allow raw accumulation to masquerade as depth.
2. Create nodes with explicit boundaries, confidence tags, and failure conditions.
3. Type edges by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark contradictions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 337 - GraphRAG and Relation Design: Failure Mode Catalogue

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is failure mode catalogue, which means the model must create nodes, type edges, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create nodes before expanding; never allow raw accumulation to masquerade as depth.
2. Type edges with explicit boundaries, confidence tags, and failure conditions.
3. Mark contradictions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Trace evidence paths through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 338 - GraphRAG and Relation Design: Update and Versioning Rule

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is update and versioning rule, which means the model must type edges, mark contradictions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Type edges before expanding; never allow raw accumulation to masquerade as depth.
2. Mark contradictions with explicit boundaries, confidence tags, and failure conditions.
3. Trace evidence paths by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create nodes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 339 - GraphRAG and Relation Design: Novelty and Hypothesis Gate

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is novelty and hypothesis gate, which means the model must mark contradictions, trace evidence paths, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mark contradictions before expanding; never allow raw accumulation to masquerade as depth.
2. Trace evidence paths with explicit boundaries, confidence tags, and failure conditions.
3. Create nodes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Type edges through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 340 - GraphRAG and Relation Design: Final Prompt Clause

This page belongs to Part 17: GraphRAG and Relation Design. Its task is to expresses the base as nodes, edges, relation types, evidence paths, and contradiction paths. The local module is final prompt clause, which means the model must trace evidence paths, create nodes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Trace evidence paths before expanding; never allow raw accumulation to masquerade as depth.
2. Create nodes with explicit boundaries, confidence tags, and failure conditions.
3. Type edges by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mark contradictions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 341 - Agentic Workflow Layer: Definition of the Operating Unit

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is definition of the operating unit, which means the model must assign agents, sequence handoffs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assign agents before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence handoffs with explicit boundaries, confidence tags, and failure conditions.
3. Define court gates by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log decisions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 342 - Agentic Workflow Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is scope boundary and non-scope guard, which means the model must sequence handoffs, define court gates, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence handoffs before expanding; never allow raw accumulation to masquerade as depth.
2. Define court gates with explicit boundaries, confidence tags, and failure conditions.
3. Log decisions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assign agents through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 343 - Agentic Workflow Layer: Kernel Extraction Move

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is kernel extraction move, which means the model must define court gates, log decisions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define court gates before expanding; never allow raw accumulation to masquerade as depth.
2. Log decisions with explicit boundaries, confidence tags, and failure conditions.
3. Assign agents by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence handoffs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 344 - Agentic Workflow Layer: Difference-Making Test

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is difference-making test, which means the model must log decisions, assign agents, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log decisions before expanding; never allow raw accumulation to masquerade as depth.
2. Assign agents with explicit boundaries, confidence tags, and failure conditions.
3. Sequence handoffs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define court gates through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 345 - Agentic Workflow Layer: Confidence and Evidence Tagging

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is confidence and evidence tagging, which means the model must assign agents, sequence handoffs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assign agents before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence handoffs with explicit boundaries, confidence tags, and failure conditions.
3. Define court gates by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log decisions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 346 - Agentic Workflow Layer: Concept Boundary Split

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is concept boundary split, which means the model must sequence handoffs, define court gates, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence handoffs before expanding; never allow raw accumulation to masquerade as depth.
2. Define court gates with explicit boundaries, confidence tags, and failure conditions.
3. Log decisions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assign agents through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 347 - Agentic Workflow Layer: Problem Deepening Method

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is problem deepening method, which means the model must define court gates, log decisions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define court gates before expanding; never allow raw accumulation to masquerade as depth.
2. Log decisions with explicit boundaries, confidence tags, and failure conditions.
3. Assign agents by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence handoffs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 348 - Agentic Workflow Layer: Skill Conversion Move

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is skill conversion move, which means the model must log decisions, assign agents, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log decisions before expanding; never allow raw accumulation to masquerade as depth.
2. Assign agents with explicit boundaries, confidence tags, and failure conditions.
3. Sequence handoffs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define court gates through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 349 - Agentic Workflow Layer: Loop Wiring Move

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is loop wiring move, which means the model must assign agents, sequence handoffs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assign agents before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence handoffs with explicit boundaries, confidence tags, and failure conditions.
3. Define court gates by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log decisions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 350 - Agentic Workflow Layer: Tool Conversion Move

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is tool conversion move, which means the model must sequence handoffs, define court gates, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence handoffs before expanding; never allow raw accumulation to masquerade as depth.
2. Define court gates with explicit boundaries, confidence tags, and failure conditions.
3. Log decisions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assign agents through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 351 - Agentic Workflow Layer: Retrieval and Metadata Move

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is retrieval and metadata move, which means the model must define court gates, log decisions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define court gates before expanding; never allow raw accumulation to masquerade as depth.
2. Log decisions with explicit boundaries, confidence tags, and failure conditions.
3. Assign agents by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence handoffs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 352 - Agentic Workflow Layer: Agent Role Assignment

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is agent role assignment, which means the model must log decisions, assign agents, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log decisions before expanding; never allow raw accumulation to masquerade as depth.
2. Assign agents with explicit boundaries, confidence tags, and failure conditions.
3. Sequence handoffs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define court gates through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 353 - Agentic Workflow Layer: Court of Destruction Pass

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is court of destruction pass, which means the model must assign agents, sequence handoffs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assign agents before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence handoffs with explicit boundaries, confidence tags, and failure conditions.
3. Define court gates by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log decisions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 354 - Agentic Workflow Layer: Repair and Reconstruction Pass

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is repair and reconstruction pass, which means the model must sequence handoffs, define court gates, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence handoffs before expanding; never allow raw accumulation to masquerade as depth.
2. Define court gates with explicit boundaries, confidence tags, and failure conditions.
3. Log decisions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assign agents through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 355 - Agentic Workflow Layer: Training and Mastery Drill

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is training and mastery drill, which means the model must define court gates, log decisions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define court gates before expanding; never allow raw accumulation to masquerade as depth.
2. Log decisions with explicit boundaries, confidence tags, and failure conditions.
3. Assign agents by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence handoffs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 356 - Agentic Workflow Layer: Output Interface Contract

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is output interface contract, which means the model must log decisions, assign agents, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log decisions before expanding; never allow raw accumulation to masquerade as depth.
2. Assign agents with explicit boundaries, confidence tags, and failure conditions.
3. Sequence handoffs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define court gates through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 357 - Agentic Workflow Layer: Failure Mode Catalogue

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is failure mode catalogue, which means the model must assign agents, sequence handoffs, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assign agents before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence handoffs with explicit boundaries, confidence tags, and failure conditions.
3. Define court gates by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log decisions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 358 - Agentic Workflow Layer: Update and Versioning Rule

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is update and versioning rule, which means the model must sequence handoffs, define court gates, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence handoffs before expanding; never allow raw accumulation to masquerade as depth.
2. Define court gates with explicit boundaries, confidence tags, and failure conditions.
3. Log decisions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assign agents through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 359 - Agentic Workflow Layer: Novelty and Hypothesis Gate

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is novelty and hypothesis gate, which means the model must define court gates, log decisions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define court gates before expanding; never allow raw accumulation to masquerade as depth.
2. Log decisions with explicit boundaries, confidence tags, and failure conditions.
3. Assign agents by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence handoffs through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 360 - Agentic Workflow Layer: Final Prompt Clause

This page belongs to Part 18: Agentic Workflow Layer. Its task is to enables multi-agent research, writing, critique, teaching, coding, product, and governance workflows. The local module is final prompt clause, which means the model must log decisions, assign agents, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log decisions before expanding; never allow raw accumulation to masquerade as depth.
2. Assign agents with explicit boundaries, confidence tags, and failure conditions.
3. Sequence handoffs by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define court gates through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 361 - Courts of Destruction: Definition of the Operating Unit

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is definition of the operating unit, which means the model must attack claims, test reduction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Attack claims before expanding; never allow raw accumulation to masquerade as depth.
2. Test reduction with explicit boundaries, confidence tags, and failure conditions.
3. Detect fake novelty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Force repair through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 362 - Courts of Destruction: Scope Boundary and Non-Scope Guard

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is scope boundary and non-scope guard, which means the model must test reduction, detect fake novelty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test reduction before expanding; never allow raw accumulation to masquerade as depth.
2. Detect fake novelty with explicit boundaries, confidence tags, and failure conditions.
3. Force repair by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Attack claims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 363 - Courts of Destruction: Kernel Extraction Move

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is kernel extraction move, which means the model must detect fake novelty, force repair, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Detect fake novelty before expanding; never allow raw accumulation to masquerade as depth.
2. Force repair with explicit boundaries, confidence tags, and failure conditions.
3. Attack claims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test reduction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 364 - Courts of Destruction: Difference-Making Test

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is difference-making test, which means the model must force repair, attack claims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Force repair before expanding; never allow raw accumulation to masquerade as depth.
2. Attack claims with explicit boundaries, confidence tags, and failure conditions.
3. Test reduction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Detect fake novelty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 365 - Courts of Destruction: Confidence and Evidence Tagging

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is confidence and evidence tagging, which means the model must attack claims, test reduction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Attack claims before expanding; never allow raw accumulation to masquerade as depth.
2. Test reduction with explicit boundaries, confidence tags, and failure conditions.
3. Detect fake novelty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Force repair through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 366 - Courts of Destruction: Concept Boundary Split

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is concept boundary split, which means the model must test reduction, detect fake novelty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test reduction before expanding; never allow raw accumulation to masquerade as depth.
2. Detect fake novelty with explicit boundaries, confidence tags, and failure conditions.
3. Force repair by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Attack claims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 367 - Courts of Destruction: Problem Deepening Method

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is problem deepening method, which means the model must detect fake novelty, force repair, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Detect fake novelty before expanding; never allow raw accumulation to masquerade as depth.
2. Force repair with explicit boundaries, confidence tags, and failure conditions.
3. Attack claims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test reduction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 368 - Courts of Destruction: Skill Conversion Move

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is skill conversion move, which means the model must force repair, attack claims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Force repair before expanding; never allow raw accumulation to masquerade as depth.
2. Attack claims with explicit boundaries, confidence tags, and failure conditions.
3. Test reduction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Detect fake novelty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 369 - Courts of Destruction: Loop Wiring Move

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is loop wiring move, which means the model must attack claims, test reduction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Attack claims before expanding; never allow raw accumulation to masquerade as depth.
2. Test reduction with explicit boundaries, confidence tags, and failure conditions.
3. Detect fake novelty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Force repair through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 370 - Courts of Destruction: Tool Conversion Move

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is tool conversion move, which means the model must test reduction, detect fake novelty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test reduction before expanding; never allow raw accumulation to masquerade as depth.
2. Detect fake novelty with explicit boundaries, confidence tags, and failure conditions.
3. Force repair by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Attack claims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 371 - Courts of Destruction: Retrieval and Metadata Move

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is retrieval and metadata move, which means the model must detect fake novelty, force repair, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Detect fake novelty before expanding; never allow raw accumulation to masquerade as depth.
2. Force repair with explicit boundaries, confidence tags, and failure conditions.
3. Attack claims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test reduction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 372 - Courts of Destruction: Agent Role Assignment

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is agent role assignment, which means the model must force repair, attack claims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Force repair before expanding; never allow raw accumulation to masquerade as depth.
2. Attack claims with explicit boundaries, confidence tags, and failure conditions.
3. Test reduction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Detect fake novelty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 373 - Courts of Destruction: Court of Destruction Pass

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is court of destruction pass, which means the model must attack claims, test reduction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Attack claims before expanding; never allow raw accumulation to masquerade as depth.
2. Test reduction with explicit boundaries, confidence tags, and failure conditions.
3. Detect fake novelty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Force repair through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 374 - Courts of Destruction: Repair and Reconstruction Pass

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is repair and reconstruction pass, which means the model must test reduction, detect fake novelty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test reduction before expanding; never allow raw accumulation to masquerade as depth.
2. Detect fake novelty with explicit boundaries, confidence tags, and failure conditions.
3. Force repair by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Attack claims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 375 - Courts of Destruction: Training and Mastery Drill

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is training and mastery drill, which means the model must detect fake novelty, force repair, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Detect fake novelty before expanding; never allow raw accumulation to masquerade as depth.
2. Force repair with explicit boundaries, confidence tags, and failure conditions.
3. Attack claims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test reduction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 376 - Courts of Destruction: Output Interface Contract

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is output interface contract, which means the model must force repair, attack claims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Force repair before expanding; never allow raw accumulation to masquerade as depth.
2. Attack claims with explicit boundaries, confidence tags, and failure conditions.
3. Test reduction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Detect fake novelty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 377 - Courts of Destruction: Failure Mode Catalogue

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is failure mode catalogue, which means the model must attack claims, test reduction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Attack claims before expanding; never allow raw accumulation to masquerade as depth.
2. Test reduction with explicit boundaries, confidence tags, and failure conditions.
3. Detect fake novelty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Force repair through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 378 - Courts of Destruction: Update and Versioning Rule

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is update and versioning rule, which means the model must test reduction, detect fake novelty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test reduction before expanding; never allow raw accumulation to masquerade as depth.
2. Detect fake novelty with explicit boundaries, confidence tags, and failure conditions.
3. Force repair by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Attack claims through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 379 - Courts of Destruction: Novelty and Hypothesis Gate

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is novelty and hypothesis gate, which means the model must detect fake novelty, force repair, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Detect fake novelty before expanding; never allow raw accumulation to masquerade as depth.
2. Force repair with explicit boundaries, confidence tags, and failure conditions.
3. Attack claims by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test reduction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 380 - Courts of Destruction: Final Prompt Clause

This page belongs to Part 19: Courts of Destruction. Its task is to subjects every output to hallucination, reduction, fake novelty, contradiction, loss, and weak-operation tests. The local module is final prompt clause, which means the model must force repair, attack claims, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Force repair before expanding; never allow raw accumulation to masquerade as depth.
2. Attack claims with explicit boundaries, confidence tags, and failure conditions.
3. Test reduction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Detect fake novelty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 381 - Evaluation Metrics Layer: Definition of the Operating Unit

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is definition of the operating unit, which means the model must score coverage, score compression, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score coverage before expanding; never allow raw accumulation to masquerade as depth.
2. Score compression with explicit boundaries, confidence tags, and failure conditions.
3. Score generativity by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score readiness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 382 - Evaluation Metrics Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is scope boundary and non-scope guard, which means the model must score compression, score generativity, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score compression before expanding; never allow raw accumulation to masquerade as depth.
2. Score generativity with explicit boundaries, confidence tags, and failure conditions.
3. Score readiness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score coverage through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 383 - Evaluation Metrics Layer: Kernel Extraction Move

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is kernel extraction move, which means the model must score generativity, score readiness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score generativity before expanding; never allow raw accumulation to masquerade as depth.
2. Score readiness with explicit boundaries, confidence tags, and failure conditions.
3. Score coverage by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score compression through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 384 - Evaluation Metrics Layer: Difference-Making Test

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is difference-making test, which means the model must score readiness, score coverage, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score readiness before expanding; never allow raw accumulation to masquerade as depth.
2. Score coverage with explicit boundaries, confidence tags, and failure conditions.
3. Score compression by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score generativity through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 385 - Evaluation Metrics Layer: Confidence and Evidence Tagging

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is confidence and evidence tagging, which means the model must score coverage, score compression, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score coverage before expanding; never allow raw accumulation to masquerade as depth.
2. Score compression with explicit boundaries, confidence tags, and failure conditions.
3. Score generativity by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score readiness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 386 - Evaluation Metrics Layer: Concept Boundary Split

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is concept boundary split, which means the model must score compression, score generativity, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score compression before expanding; never allow raw accumulation to masquerade as depth.
2. Score generativity with explicit boundaries, confidence tags, and failure conditions.
3. Score readiness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score coverage through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 387 - Evaluation Metrics Layer: Problem Deepening Method

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is problem deepening method, which means the model must score generativity, score readiness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score generativity before expanding; never allow raw accumulation to masquerade as depth.
2. Score readiness with explicit boundaries, confidence tags, and failure conditions.
3. Score coverage by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score compression through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 388 - Evaluation Metrics Layer: Skill Conversion Move

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is skill conversion move, which means the model must score readiness, score coverage, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score readiness before expanding; never allow raw accumulation to masquerade as depth.
2. Score coverage with explicit boundaries, confidence tags, and failure conditions.
3. Score compression by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score generativity through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 389 - Evaluation Metrics Layer: Loop Wiring Move

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is loop wiring move, which means the model must score coverage, score compression, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score coverage before expanding; never allow raw accumulation to masquerade as depth.
2. Score compression with explicit boundaries, confidence tags, and failure conditions.
3. Score generativity by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score readiness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 390 - Evaluation Metrics Layer: Tool Conversion Move

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is tool conversion move, which means the model must score compression, score generativity, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score compression before expanding; never allow raw accumulation to masquerade as depth.
2. Score generativity with explicit boundaries, confidence tags, and failure conditions.
3. Score readiness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score coverage through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 391 - Evaluation Metrics Layer: Retrieval and Metadata Move

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is retrieval and metadata move, which means the model must score generativity, score readiness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score generativity before expanding; never allow raw accumulation to masquerade as depth.
2. Score readiness with explicit boundaries, confidence tags, and failure conditions.
3. Score coverage by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score compression through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 392 - Evaluation Metrics Layer: Agent Role Assignment

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is agent role assignment, which means the model must score readiness, score coverage, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score readiness before expanding; never allow raw accumulation to masquerade as depth.
2. Score coverage with explicit boundaries, confidence tags, and failure conditions.
3. Score compression by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score generativity through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 393 - Evaluation Metrics Layer: Court of Destruction Pass

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is court of destruction pass, which means the model must score coverage, score compression, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score coverage before expanding; never allow raw accumulation to masquerade as depth.
2. Score compression with explicit boundaries, confidence tags, and failure conditions.
3. Score generativity by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score readiness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 394 - Evaluation Metrics Layer: Repair and Reconstruction Pass

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is repair and reconstruction pass, which means the model must score compression, score generativity, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score compression before expanding; never allow raw accumulation to masquerade as depth.
2. Score generativity with explicit boundaries, confidence tags, and failure conditions.
3. Score readiness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score coverage through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 395 - Evaluation Metrics Layer: Training and Mastery Drill

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is training and mastery drill, which means the model must score generativity, score readiness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score generativity before expanding; never allow raw accumulation to masquerade as depth.
2. Score readiness with explicit boundaries, confidence tags, and failure conditions.
3. Score coverage by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score compression through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 396 - Evaluation Metrics Layer: Output Interface Contract

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is output interface contract, which means the model must score readiness, score coverage, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score readiness before expanding; never allow raw accumulation to masquerade as depth.
2. Score coverage with explicit boundaries, confidence tags, and failure conditions.
3. Score compression by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score generativity through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 397 - Evaluation Metrics Layer: Failure Mode Catalogue

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is failure mode catalogue, which means the model must score coverage, score compression, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score coverage before expanding; never allow raw accumulation to masquerade as depth.
2. Score compression with explicit boundaries, confidence tags, and failure conditions.
3. Score generativity by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score readiness through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 398 - Evaluation Metrics Layer: Update and Versioning Rule

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is update and versioning rule, which means the model must score compression, score generativity, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score compression before expanding; never allow raw accumulation to masquerade as depth.
2. Score generativity with explicit boundaries, confidence tags, and failure conditions.
3. Score readiness by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score coverage through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 399 - Evaluation Metrics Layer: Novelty and Hypothesis Gate

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is novelty and hypothesis gate, which means the model must score generativity, score readiness, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score generativity before expanding; never allow raw accumulation to masquerade as depth.
2. Score readiness with explicit boundaries, confidence tags, and failure conditions.
3. Score coverage by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score compression through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 400 - Evaluation Metrics Layer: Final Prompt Clause

This page belongs to Part 20: Evaluation Metrics Layer. Its task is to defines how to judge coverage, compression, generativity, reliability, skillfulness, and OS readiness. The local module is final prompt clause, which means the model must score readiness, score coverage, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Score readiness before expanding; never allow raw accumulation to masquerade as depth.
2. Score coverage with explicit boundaries, confidence tags, and failure conditions.
3. Score compression by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Score generativity through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 401 - Science and Technology Adapter: Definition of the Operating Unit

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is definition of the operating unit, which means the model must separate model from implementation, record assumptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate model from implementation before expanding; never allow raw accumulation to masquerade as depth.
2. Record assumptions with explicit boundaries, confidence tags, and failure conditions.
3. Define experiments by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track benchmarks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 402 - Science and Technology Adapter: Scope Boundary and Non-Scope Guard

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is scope boundary and non-scope guard, which means the model must record assumptions, define experiments, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record assumptions before expanding; never allow raw accumulation to masquerade as depth.
2. Define experiments with explicit boundaries, confidence tags, and failure conditions.
3. Track benchmarks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate model from implementation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 403 - Science and Technology Adapter: Kernel Extraction Move

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is kernel extraction move, which means the model must define experiments, track benchmarks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define experiments before expanding; never allow raw accumulation to masquerade as depth.
2. Track benchmarks with explicit boundaries, confidence tags, and failure conditions.
3. Separate model from implementation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record assumptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 404 - Science and Technology Adapter: Difference-Making Test

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is difference-making test, which means the model must track benchmarks, separate model from implementation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track benchmarks before expanding; never allow raw accumulation to masquerade as depth.
2. Separate model from implementation with explicit boundaries, confidence tags, and failure conditions.
3. Record assumptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define experiments through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 405 - Science and Technology Adapter: Confidence and Evidence Tagging

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is confidence and evidence tagging, which means the model must separate model from implementation, record assumptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate model from implementation before expanding; never allow raw accumulation to masquerade as depth.
2. Record assumptions with explicit boundaries, confidence tags, and failure conditions.
3. Define experiments by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track benchmarks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 406 - Science and Technology Adapter: Concept Boundary Split

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is concept boundary split, which means the model must record assumptions, define experiments, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record assumptions before expanding; never allow raw accumulation to masquerade as depth.
2. Define experiments with explicit boundaries, confidence tags, and failure conditions.
3. Track benchmarks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate model from implementation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 407 - Science and Technology Adapter: Problem Deepening Method

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is problem deepening method, which means the model must define experiments, track benchmarks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define experiments before expanding; never allow raw accumulation to masquerade as depth.
2. Track benchmarks with explicit boundaries, confidence tags, and failure conditions.
3. Separate model from implementation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record assumptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 408 - Science and Technology Adapter: Skill Conversion Move

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is skill conversion move, which means the model must track benchmarks, separate model from implementation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track benchmarks before expanding; never allow raw accumulation to masquerade as depth.
2. Separate model from implementation with explicit boundaries, confidence tags, and failure conditions.
3. Record assumptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define experiments through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 409 - Science and Technology Adapter: Loop Wiring Move

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is loop wiring move, which means the model must separate model from implementation, record assumptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate model from implementation before expanding; never allow raw accumulation to masquerade as depth.
2. Record assumptions with explicit boundaries, confidence tags, and failure conditions.
3. Define experiments by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track benchmarks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 410 - Science and Technology Adapter: Tool Conversion Move

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is tool conversion move, which means the model must record assumptions, define experiments, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record assumptions before expanding; never allow raw accumulation to masquerade as depth.
2. Define experiments with explicit boundaries, confidence tags, and failure conditions.
3. Track benchmarks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate model from implementation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 411 - Science and Technology Adapter: Retrieval and Metadata Move

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is retrieval and metadata move, which means the model must define experiments, track benchmarks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define experiments before expanding; never allow raw accumulation to masquerade as depth.
2. Track benchmarks with explicit boundaries, confidence tags, and failure conditions.
3. Separate model from implementation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record assumptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 412 - Science and Technology Adapter: Agent Role Assignment

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is agent role assignment, which means the model must track benchmarks, separate model from implementation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track benchmarks before expanding; never allow raw accumulation to masquerade as depth.
2. Separate model from implementation with explicit boundaries, confidence tags, and failure conditions.
3. Record assumptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define experiments through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 413 - Science and Technology Adapter: Court of Destruction Pass

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is court of destruction pass, which means the model must separate model from implementation, record assumptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate model from implementation before expanding; never allow raw accumulation to masquerade as depth.
2. Record assumptions with explicit boundaries, confidence tags, and failure conditions.
3. Define experiments by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track benchmarks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 414 - Science and Technology Adapter: Repair and Reconstruction Pass

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is repair and reconstruction pass, which means the model must record assumptions, define experiments, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record assumptions before expanding; never allow raw accumulation to masquerade as depth.
2. Define experiments with explicit boundaries, confidence tags, and failure conditions.
3. Track benchmarks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate model from implementation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 415 - Science and Technology Adapter: Training and Mastery Drill

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is training and mastery drill, which means the model must define experiments, track benchmarks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define experiments before expanding; never allow raw accumulation to masquerade as depth.
2. Track benchmarks with explicit boundaries, confidence tags, and failure conditions.
3. Separate model from implementation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record assumptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 416 - Science and Technology Adapter: Output Interface Contract

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is output interface contract, which means the model must track benchmarks, separate model from implementation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track benchmarks before expanding; never allow raw accumulation to masquerade as depth.
2. Separate model from implementation with explicit boundaries, confidence tags, and failure conditions.
3. Record assumptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define experiments through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 417 - Science and Technology Adapter: Failure Mode Catalogue

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is failure mode catalogue, which means the model must separate model from implementation, record assumptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate model from implementation before expanding; never allow raw accumulation to masquerade as depth.
2. Record assumptions with explicit boundaries, confidence tags, and failure conditions.
3. Define experiments by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Track benchmarks through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 418 - Science and Technology Adapter: Update and Versioning Rule

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is update and versioning rule, which means the model must record assumptions, define experiments, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record assumptions before expanding; never allow raw accumulation to masquerade as depth.
2. Define experiments with explicit boundaries, confidence tags, and failure conditions.
3. Track benchmarks by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate model from implementation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 419 - Science and Technology Adapter: Novelty and Hypothesis Gate

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is novelty and hypothesis gate, which means the model must define experiments, track benchmarks, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define experiments before expanding; never allow raw accumulation to masquerade as depth.
2. Track benchmarks with explicit boundaries, confidence tags, and failure conditions.
3. Separate model from implementation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record assumptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 420 - Science and Technology Adapter: Final Prompt Clause

This page belongs to Part 21: Science and Technology Adapter. Its task is to adapts the generic architecture to scientific, technical, engineering, AI, and software domains. The local module is final prompt clause, which means the model must track benchmarks, separate model from implementation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Track benchmarks before expanding; never allow raw accumulation to masquerade as depth.
2. Separate model from implementation with explicit boundaries, confidence tags, and failure conditions.
3. Record assumptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define experiments through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 421 - Humanities, History, and Theology Adapter: Definition of the Operating Unit

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is definition of the operating unit, which means the model must separate text from interpretation, rank evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate text from interpretation before expanding; never allow raw accumulation to masquerade as depth.
2. Rank evidence with explicit boundaries, confidence tags, and failure conditions.
3. Preserve schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid partisan flattening through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 422 - Humanities, History, and Theology Adapter: Scope Boundary and Non-Scope Guard

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is scope boundary and non-scope guard, which means the model must rank evidence, preserve schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve schools with explicit boundaries, confidence tags, and failure conditions.
3. Avoid partisan flattening by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate text from interpretation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 423 - Humanities, History, and Theology Adapter: Kernel Extraction Move

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is kernel extraction move, which means the model must preserve schools, avoid partisan flattening, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve schools before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid partisan flattening with explicit boundaries, confidence tags, and failure conditions.
3. Separate text from interpretation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 424 - Humanities, History, and Theology Adapter: Difference-Making Test

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is difference-making test, which means the model must avoid partisan flattening, separate text from interpretation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid partisan flattening before expanding; never allow raw accumulation to masquerade as depth.
2. Separate text from interpretation with explicit boundaries, confidence tags, and failure conditions.
3. Rank evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 425 - Humanities, History, and Theology Adapter: Confidence and Evidence Tagging

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is confidence and evidence tagging, which means the model must separate text from interpretation, rank evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate text from interpretation before expanding; never allow raw accumulation to masquerade as depth.
2. Rank evidence with explicit boundaries, confidence tags, and failure conditions.
3. Preserve schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid partisan flattening through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 426 - Humanities, History, and Theology Adapter: Concept Boundary Split

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is concept boundary split, which means the model must rank evidence, preserve schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve schools with explicit boundaries, confidence tags, and failure conditions.
3. Avoid partisan flattening by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate text from interpretation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 427 - Humanities, History, and Theology Adapter: Problem Deepening Method

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is problem deepening method, which means the model must preserve schools, avoid partisan flattening, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve schools before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid partisan flattening with explicit boundaries, confidence tags, and failure conditions.
3. Separate text from interpretation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 428 - Humanities, History, and Theology Adapter: Skill Conversion Move

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is skill conversion move, which means the model must avoid partisan flattening, separate text from interpretation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid partisan flattening before expanding; never allow raw accumulation to masquerade as depth.
2. Separate text from interpretation with explicit boundaries, confidence tags, and failure conditions.
3. Rank evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 429 - Humanities, History, and Theology Adapter: Loop Wiring Move

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is loop wiring move, which means the model must separate text from interpretation, rank evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate text from interpretation before expanding; never allow raw accumulation to masquerade as depth.
2. Rank evidence with explicit boundaries, confidence tags, and failure conditions.
3. Preserve schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid partisan flattening through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 430 - Humanities, History, and Theology Adapter: Tool Conversion Move

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is tool conversion move, which means the model must rank evidence, preserve schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve schools with explicit boundaries, confidence tags, and failure conditions.
3. Avoid partisan flattening by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate text from interpretation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 431 - Humanities, History, and Theology Adapter: Retrieval and Metadata Move

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is retrieval and metadata move, which means the model must preserve schools, avoid partisan flattening, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve schools before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid partisan flattening with explicit boundaries, confidence tags, and failure conditions.
3. Separate text from interpretation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 432 - Humanities, History, and Theology Adapter: Agent Role Assignment

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is agent role assignment, which means the model must avoid partisan flattening, separate text from interpretation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid partisan flattening before expanding; never allow raw accumulation to masquerade as depth.
2. Separate text from interpretation with explicit boundaries, confidence tags, and failure conditions.
3. Rank evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 433 - Humanities, History, and Theology Adapter: Court of Destruction Pass

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is court of destruction pass, which means the model must separate text from interpretation, rank evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate text from interpretation before expanding; never allow raw accumulation to masquerade as depth.
2. Rank evidence with explicit boundaries, confidence tags, and failure conditions.
3. Preserve schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid partisan flattening through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 434 - Humanities, History, and Theology Adapter: Repair and Reconstruction Pass

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is repair and reconstruction pass, which means the model must rank evidence, preserve schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve schools with explicit boundaries, confidence tags, and failure conditions.
3. Avoid partisan flattening by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate text from interpretation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 435 - Humanities, History, and Theology Adapter: Training and Mastery Drill

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is training and mastery drill, which means the model must preserve schools, avoid partisan flattening, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve schools before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid partisan flattening with explicit boundaries, confidence tags, and failure conditions.
3. Separate text from interpretation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 436 - Humanities, History, and Theology Adapter: Output Interface Contract

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is output interface contract, which means the model must avoid partisan flattening, separate text from interpretation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid partisan flattening before expanding; never allow raw accumulation to masquerade as depth.
2. Separate text from interpretation with explicit boundaries, confidence tags, and failure conditions.
3. Rank evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 437 - Humanities, History, and Theology Adapter: Failure Mode Catalogue

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is failure mode catalogue, which means the model must separate text from interpretation, rank evidence, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Separate text from interpretation before expanding; never allow raw accumulation to masquerade as depth.
2. Rank evidence with explicit boundaries, confidence tags, and failure conditions.
3. Preserve schools by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Avoid partisan flattening through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 438 - Humanities, History, and Theology Adapter: Update and Versioning Rule

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is update and versioning rule, which means the model must rank evidence, preserve schools, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rank evidence before expanding; never allow raw accumulation to masquerade as depth.
2. Preserve schools with explicit boundaries, confidence tags, and failure conditions.
3. Avoid partisan flattening by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Separate text from interpretation through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 439 - Humanities, History, and Theology Adapter: Novelty and Hypothesis Gate

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is novelty and hypothesis gate, which means the model must preserve schools, avoid partisan flattening, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Preserve schools before expanding; never allow raw accumulation to masquerade as depth.
2. Avoid partisan flattening with explicit boundaries, confidence tags, and failure conditions.
3. Separate text from interpretation by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rank evidence through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 440 - Humanities, History, and Theology Adapter: Final Prompt Clause

This page belongs to Part 22: Humanities, History, and Theology Adapter. Its task is to adapts the architecture to interpretive, textual, historical, philosophical, and religious domains. The local module is final prompt clause, which means the model must avoid partisan flattening, separate text from interpretation, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Avoid partisan flattening before expanding; never allow raw accumulation to masquerade as depth.
2. Separate text from interpretation with explicit boundaries, confidence tags, and failure conditions.
3. Rank evidence by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Preserve schools through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 441 - Education and Training Adapter: Definition of the Operating Unit

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is definition of the operating unit, which means the model must sequence mastery, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must index the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence mastery before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Test transfer by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Repair misconceptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 442 - Education and Training Adapter: Scope Boundary and Non-Scope Guard

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is scope boundary and non-scope guard, which means the model must write drills, test transfer, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must train the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Test transfer with explicit boundaries, confidence tags, and failure conditions.
3. Repair misconceptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence mastery through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 443 - Education and Training Adapter: Kernel Extraction Move

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is kernel extraction move, which means the model must test transfer, repair misconceptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must extract the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test transfer before expanding; never allow raw accumulation to masquerade as depth.
2. Repair misconceptions with explicit boundaries, confidence tags, and failure conditions.
3. Sequence mastery by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 444 - Education and Training Adapter: Difference-Making Test

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is difference-making test, which means the model must repair misconceptions, sequence mastery, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must compress the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Repair misconceptions before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence mastery with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test transfer through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 445 - Education and Training Adapter: Confidence and Evidence Tagging

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is confidence and evidence tagging, which means the model must sequence mastery, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must map the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence mastery before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Test transfer by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Repair misconceptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 446 - Education and Training Adapter: Concept Boundary Split

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is concept boundary split, which means the model must write drills, test transfer, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must route the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Test transfer with explicit boundaries, confidence tags, and failure conditions.
3. Repair misconceptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence mastery through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 447 - Education and Training Adapter: Problem Deepening Method

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is problem deepening method, which means the model must test transfer, repair misconceptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must score the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test transfer before expanding; never allow raw accumulation to masquerade as depth.
2. Repair misconceptions with explicit boundaries, confidence tags, and failure conditions.
3. Sequence mastery by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 448 - Education and Training Adapter: Skill Conversion Move

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is skill conversion move, which means the model must repair misconceptions, sequence mastery, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must separate the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Repair misconceptions before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence mastery with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test transfer through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 449 - Education and Training Adapter: Loop Wiring Move

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is loop wiring move, which means the model must sequence mastery, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must convert the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence mastery before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Test transfer by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Repair misconceptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 450 - Education and Training Adapter: Tool Conversion Move

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is tool conversion move, which means the model must write drills, test transfer, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must test the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Test transfer with explicit boundaries, confidence tags, and failure conditions.
3. Repair misconceptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence mastery through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 451 - Education and Training Adapter: Retrieval and Metadata Move

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is retrieval and metadata move, which means the model must test transfer, repair misconceptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must repair the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test transfer before expanding; never allow raw accumulation to masquerade as depth.
2. Repair misconceptions with explicit boundaries, confidence tags, and failure conditions.
3. Sequence mastery by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 452 - Education and Training Adapter: Agent Role Assignment

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is agent role assignment, which means the model must repair misconceptions, sequence mastery, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must upgrade the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Repair misconceptions before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence mastery with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test transfer through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 453 - Education and Training Adapter: Court of Destruction Pass

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is court of destruction pass, which means the model must sequence mastery, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must index the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence mastery before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Test transfer by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Repair misconceptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 454 - Education and Training Adapter: Repair and Reconstruction Pass

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is repair and reconstruction pass, which means the model must write drills, test transfer, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must train the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Test transfer with explicit boundaries, confidence tags, and failure conditions.
3. Repair misconceptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence mastery through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 455 - Education and Training Adapter: Training and Mastery Drill

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is training and mastery drill, which means the model must test transfer, repair misconceptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must extract the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test transfer before expanding; never allow raw accumulation to masquerade as depth.
2. Repair misconceptions with explicit boundaries, confidence tags, and failure conditions.
3. Sequence mastery by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 456 - Education and Training Adapter: Output Interface Contract

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is output interface contract, which means the model must repair misconceptions, sequence mastery, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must compress the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Repair misconceptions before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence mastery with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test transfer through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 457 - Education and Training Adapter: Failure Mode Catalogue

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is failure mode catalogue, which means the model must sequence mastery, write drills, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must map the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Sequence mastery before expanding; never allow raw accumulation to masquerade as depth.
2. Write drills with explicit boundaries, confidence tags, and failure conditions.
3. Test transfer by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Repair misconceptions through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 458 - Education and Training Adapter: Update and Versioning Rule

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is update and versioning rule, which means the model must write drills, test transfer, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must route the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write drills before expanding; never allow raw accumulation to masquerade as depth.
2. Test transfer with explicit boundaries, confidence tags, and failure conditions.
3. Repair misconceptions by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Sequence mastery through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 459 - Education and Training Adapter: Novelty and Hypothesis Gate

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is novelty and hypothesis gate, which means the model must test transfer, repair misconceptions, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must score the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test transfer before expanding; never allow raw accumulation to masquerade as depth.
2. Repair misconceptions with explicit boundaries, confidence tags, and failure conditions.
3. Sequence mastery by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write drills through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 460 - Education and Training Adapter: Final Prompt Clause

This page belongs to Part 23: Education and Training Adapter. Its task is to turns the base into curricula, drills, mastery paths, tests, and expert apprenticeship loops. The local module is final prompt clause, which means the model must repair misconceptions, sequence mastery, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must separate the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Repair misconceptions before expanding; never allow raw accumulation to masquerade as depth.
2. Sequence mastery with explicit boundaries, confidence tags, and failure conditions.
3. Write drills by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test transfer through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 461 - Research Writing and Publishing Adapter: Definition of the Operating Unit

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is definition of the operating unit, which means the model must build thesis spine, map literature, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must train the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build thesis spine before expanding; never allow raw accumulation to masquerade as depth.
2. Map literature with explicit boundaries, confidence tags, and failure conditions.
3. Write objections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State limits through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 462 - Research Writing and Publishing Adapter: Scope Boundary and Non-Scope Guard

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is scope boundary and non-scope guard, which means the model must map literature, write objections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must extract the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map literature before expanding; never allow raw accumulation to masquerade as depth.
2. Write objections with explicit boundaries, confidence tags, and failure conditions.
3. State limits by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build thesis spine through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 463 - Research Writing and Publishing Adapter: Kernel Extraction Move

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is kernel extraction move, which means the model must write objections, state limits, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must compress the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write objections before expanding; never allow raw accumulation to masquerade as depth.
2. State limits with explicit boundaries, confidence tags, and failure conditions.
3. Build thesis spine by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map literature through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 464 - Research Writing and Publishing Adapter: Difference-Making Test

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is difference-making test, which means the model must state limits, build thesis spine, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must map the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State limits before expanding; never allow raw accumulation to masquerade as depth.
2. Build thesis spine with explicit boundaries, confidence tags, and failure conditions.
3. Map literature by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write objections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 465 - Research Writing and Publishing Adapter: Confidence and Evidence Tagging

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is confidence and evidence tagging, which means the model must build thesis spine, map literature, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must route the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build thesis spine before expanding; never allow raw accumulation to masquerade as depth.
2. Map literature with explicit boundaries, confidence tags, and failure conditions.
3. Write objections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State limits through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 466 - Research Writing and Publishing Adapter: Concept Boundary Split

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is concept boundary split, which means the model must map literature, write objections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must score the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map literature before expanding; never allow raw accumulation to masquerade as depth.
2. Write objections with explicit boundaries, confidence tags, and failure conditions.
3. State limits by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build thesis spine through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 467 - Research Writing and Publishing Adapter: Problem Deepening Method

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is problem deepening method, which means the model must write objections, state limits, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must separate the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write objections before expanding; never allow raw accumulation to masquerade as depth.
2. State limits with explicit boundaries, confidence tags, and failure conditions.
3. Build thesis spine by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map literature through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 468 - Research Writing and Publishing Adapter: Skill Conversion Move

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is skill conversion move, which means the model must state limits, build thesis spine, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must convert the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State limits before expanding; never allow raw accumulation to masquerade as depth.
2. Build thesis spine with explicit boundaries, confidence tags, and failure conditions.
3. Map literature by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write objections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 469 - Research Writing and Publishing Adapter: Loop Wiring Move

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is loop wiring move, which means the model must build thesis spine, map literature, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must test the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build thesis spine before expanding; never allow raw accumulation to masquerade as depth.
2. Map literature with explicit boundaries, confidence tags, and failure conditions.
3. Write objections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State limits through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 470 - Research Writing and Publishing Adapter: Tool Conversion Move

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is tool conversion move, which means the model must map literature, write objections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must repair the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map literature before expanding; never allow raw accumulation to masquerade as depth.
2. Write objections with explicit boundaries, confidence tags, and failure conditions.
3. State limits by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build thesis spine through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 471 - Research Writing and Publishing Adapter: Retrieval and Metadata Move

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is retrieval and metadata move, which means the model must write objections, state limits, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must upgrade the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write objections before expanding; never allow raw accumulation to masquerade as depth.
2. State limits with explicit boundaries, confidence tags, and failure conditions.
3. Build thesis spine by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map literature through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 472 - Research Writing and Publishing Adapter: Agent Role Assignment

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is agent role assignment, which means the model must state limits, build thesis spine, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must index the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State limits before expanding; never allow raw accumulation to masquerade as depth.
2. Build thesis spine with explicit boundaries, confidence tags, and failure conditions.
3. Map literature by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write objections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 473 - Research Writing and Publishing Adapter: Court of Destruction Pass

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is court of destruction pass, which means the model must build thesis spine, map literature, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must train the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build thesis spine before expanding; never allow raw accumulation to masquerade as depth.
2. Map literature with explicit boundaries, confidence tags, and failure conditions.
3. Write objections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State limits through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 474 - Research Writing and Publishing Adapter: Repair and Reconstruction Pass

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is repair and reconstruction pass, which means the model must map literature, write objections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must extract the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map literature before expanding; never allow raw accumulation to masquerade as depth.
2. Write objections with explicit boundaries, confidence tags, and failure conditions.
3. State limits by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build thesis spine through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 475 - Research Writing and Publishing Adapter: Training and Mastery Drill

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is training and mastery drill, which means the model must write objections, state limits, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must compress the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write objections before expanding; never allow raw accumulation to masquerade as depth.
2. State limits with explicit boundaries, confidence tags, and failure conditions.
3. Build thesis spine by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map literature through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 476 - Research Writing and Publishing Adapter: Output Interface Contract

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is output interface contract, which means the model must state limits, build thesis spine, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must map the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State limits before expanding; never allow raw accumulation to masquerade as depth.
2. Build thesis spine with explicit boundaries, confidence tags, and failure conditions.
3. Map literature by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write objections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 477 - Research Writing and Publishing Adapter: Failure Mode Catalogue

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is failure mode catalogue, which means the model must build thesis spine, map literature, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must route the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build thesis spine before expanding; never allow raw accumulation to masquerade as depth.
2. Map literature with explicit boundaries, confidence tags, and failure conditions.
3. Write objections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. State limits through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 478 - Research Writing and Publishing Adapter: Update and Versioning Rule

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is update and versioning rule, which means the model must map literature, write objections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must score the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map literature before expanding; never allow raw accumulation to masquerade as depth.
2. Write objections with explicit boundaries, confidence tags, and failure conditions.
3. State limits by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build thesis spine through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 479 - Research Writing and Publishing Adapter: Novelty and Hypothesis Gate

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is novelty and hypothesis gate, which means the model must write objections, state limits, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must separate the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write objections before expanding; never allow raw accumulation to masquerade as depth.
2. State limits with explicit boundaries, confidence tags, and failure conditions.
3. Build thesis spine by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map literature through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 480 - Research Writing and Publishing Adapter: Final Prompt Clause

This page belongs to Part 24: Research Writing and Publishing Adapter. Its task is to converts the base into papers, books, literature maps, theses, arguments, and revision systems. The local module is final prompt clause, which means the model must state limits, build thesis spine, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must convert the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. State limits before expanding; never allow raw accumulation to masquerade as depth.
2. Build thesis spine with explicit boundaries, confidence tags, and failure conditions.
3. Map literature by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write objections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 481 - Product and Business Adapter: Definition of the Operating Unit

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is definition of the operating unit, which means the model must identify user value, map market risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Identify user value before expanding; never allow raw accumulation to masquerade as depth.
2. Map market risk with explicit boundaries, confidence tags, and failure conditions.
3. Define operations by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure traction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 482 - Product and Business Adapter: Scope Boundary and Non-Scope Guard

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is scope boundary and non-scope guard, which means the model must map market risk, define operations, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map market risk before expanding; never allow raw accumulation to masquerade as depth.
2. Define operations with explicit boundaries, confidence tags, and failure conditions.
3. Measure traction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Identify user value through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 483 - Product and Business Adapter: Kernel Extraction Move

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is kernel extraction move, which means the model must define operations, measure traction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define operations before expanding; never allow raw accumulation to masquerade as depth.
2. Measure traction with explicit boundaries, confidence tags, and failure conditions.
3. Identify user value by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map market risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 484 - Product and Business Adapter: Difference-Making Test

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is difference-making test, which means the model must measure traction, identify user value, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure traction before expanding; never allow raw accumulation to masquerade as depth.
2. Identify user value with explicit boundaries, confidence tags, and failure conditions.
3. Map market risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define operations through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 485 - Product and Business Adapter: Confidence and Evidence Tagging

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is confidence and evidence tagging, which means the model must identify user value, map market risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Identify user value before expanding; never allow raw accumulation to masquerade as depth.
2. Map market risk with explicit boundaries, confidence tags, and failure conditions.
3. Define operations by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure traction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 486 - Product and Business Adapter: Concept Boundary Split

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is concept boundary split, which means the model must map market risk, define operations, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map market risk before expanding; never allow raw accumulation to masquerade as depth.
2. Define operations with explicit boundaries, confidence tags, and failure conditions.
3. Measure traction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Identify user value through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 487 - Product and Business Adapter: Problem Deepening Method

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is problem deepening method, which means the model must define operations, measure traction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define operations before expanding; never allow raw accumulation to masquerade as depth.
2. Measure traction with explicit boundaries, confidence tags, and failure conditions.
3. Identify user value by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map market risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 488 - Product and Business Adapter: Skill Conversion Move

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is skill conversion move, which means the model must measure traction, identify user value, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure traction before expanding; never allow raw accumulation to masquerade as depth.
2. Identify user value with explicit boundaries, confidence tags, and failure conditions.
3. Map market risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define operations through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 489 - Product and Business Adapter: Loop Wiring Move

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is loop wiring move, which means the model must identify user value, map market risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must repair the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Identify user value before expanding; never allow raw accumulation to masquerade as depth.
2. Map market risk with explicit boundaries, confidence tags, and failure conditions.
3. Define operations by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure traction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 490 - Product and Business Adapter: Tool Conversion Move

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is tool conversion move, which means the model must map market risk, define operations, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must upgrade the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map market risk before expanding; never allow raw accumulation to masquerade as depth.
2. Define operations with explicit boundaries, confidence tags, and failure conditions.
3. Measure traction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Identify user value through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 491 - Product and Business Adapter: Retrieval and Metadata Move

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is retrieval and metadata move, which means the model must define operations, measure traction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must index the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define operations before expanding; never allow raw accumulation to masquerade as depth.
2. Measure traction with explicit boundaries, confidence tags, and failure conditions.
3. Identify user value by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map market risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 492 - Product and Business Adapter: Agent Role Assignment

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is agent role assignment, which means the model must measure traction, identify user value, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must train the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure traction before expanding; never allow raw accumulation to masquerade as depth.
2. Identify user value with explicit boundaries, confidence tags, and failure conditions.
3. Map market risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define operations through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 493 - Product and Business Adapter: Court of Destruction Pass

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is court of destruction pass, which means the model must identify user value, map market risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must extract the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Identify user value before expanding; never allow raw accumulation to masquerade as depth.
2. Map market risk with explicit boundaries, confidence tags, and failure conditions.
3. Define operations by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure traction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 494 - Product and Business Adapter: Repair and Reconstruction Pass

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is repair and reconstruction pass, which means the model must map market risk, define operations, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must compress the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map market risk before expanding; never allow raw accumulation to masquerade as depth.
2. Define operations with explicit boundaries, confidence tags, and failure conditions.
3. Measure traction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Identify user value through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 495 - Product and Business Adapter: Training and Mastery Drill

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is training and mastery drill, which means the model must define operations, measure traction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must map the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define operations before expanding; never allow raw accumulation to masquerade as depth.
2. Measure traction with explicit boundaries, confidence tags, and failure conditions.
3. Identify user value by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map market risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 496 - Product and Business Adapter: Output Interface Contract

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is output interface contract, which means the model must measure traction, identify user value, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must route the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure traction before expanding; never allow raw accumulation to masquerade as depth.
2. Identify user value with explicit boundaries, confidence tags, and failure conditions.
3. Map market risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define operations through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 497 - Product and Business Adapter: Failure Mode Catalogue

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is failure mode catalogue, which means the model must identify user value, map market risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must score the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Identify user value before expanding; never allow raw accumulation to masquerade as depth.
2. Map market risk with explicit boundaries, confidence tags, and failure conditions.
3. Define operations by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Measure traction through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 498 - Product and Business Adapter: Update and Versioning Rule

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is update and versioning rule, which means the model must map market risk, define operations, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must separate the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Map market risk before expanding; never allow raw accumulation to masquerade as depth.
2. Define operations with explicit boundaries, confidence tags, and failure conditions.
3. Measure traction by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Identify user value through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 499 - Product and Business Adapter: Novelty and Hypothesis Gate

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is novelty and hypothesis gate, which means the model must define operations, measure traction, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must convert the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Define operations before expanding; never allow raw accumulation to masquerade as depth.
2. Measure traction with explicit boundaries, confidence tags, and failure conditions.
3. Identify user value by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Map market risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 500 - Product and Business Adapter: Final Prompt Clause

This page belongs to Part 25: Product and Business Adapter. Its task is to translates knowledge into market decisions, product systems, offers, operations, and strategic experiments. The local module is final prompt clause, which means the model must measure traction, identify user value, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must test the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Measure traction before expanding; never allow raw accumulation to masquerade as depth.
2. Identify user value with explicit boundaries, confidence tags, and failure conditions.
3. Map market risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Define operations through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 501 - Innovation and Novelty Engine: Definition of the Operating Unit

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is definition of the operating unit, which means the model must mine gaps, combine distant branches, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mine gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Combine distant branches with explicit boundaries, confidence tags, and failure conditions.
3. Build prototypes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test novelty function through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 502 - Innovation and Novelty Engine: Scope Boundary and Non-Scope Guard

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is scope boundary and non-scope guard, which means the model must combine distant branches, build prototypes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Combine distant branches before expanding; never allow raw accumulation to masquerade as depth.
2. Build prototypes with explicit boundaries, confidence tags, and failure conditions.
3. Test novelty function by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mine gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 503 - Innovation and Novelty Engine: Kernel Extraction Move

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is kernel extraction move, which means the model must build prototypes, test novelty function, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build prototypes before expanding; never allow raw accumulation to masquerade as depth.
2. Test novelty function with explicit boundaries, confidence tags, and failure conditions.
3. Mine gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Combine distant branches through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 504 - Innovation and Novelty Engine: Difference-Making Test

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is difference-making test, which means the model must test novelty function, mine gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test novelty function before expanding; never allow raw accumulation to masquerade as depth.
2. Mine gaps with explicit boundaries, confidence tags, and failure conditions.
3. Combine distant branches by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build prototypes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 505 - Innovation and Novelty Engine: Confidence and Evidence Tagging

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is confidence and evidence tagging, which means the model must mine gaps, combine distant branches, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mine gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Combine distant branches with explicit boundaries, confidence tags, and failure conditions.
3. Build prototypes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test novelty function through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 506 - Innovation and Novelty Engine: Concept Boundary Split

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is concept boundary split, which means the model must combine distant branches, build prototypes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Combine distant branches before expanding; never allow raw accumulation to masquerade as depth.
2. Build prototypes with explicit boundaries, confidence tags, and failure conditions.
3. Test novelty function by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mine gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 507 - Innovation and Novelty Engine: Problem Deepening Method

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is problem deepening method, which means the model must build prototypes, test novelty function, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build prototypes before expanding; never allow raw accumulation to masquerade as depth.
2. Test novelty function with explicit boundaries, confidence tags, and failure conditions.
3. Mine gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Combine distant branches through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 508 - Innovation and Novelty Engine: Skill Conversion Move

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is skill conversion move, which means the model must test novelty function, mine gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test novelty function before expanding; never allow raw accumulation to masquerade as depth.
2. Mine gaps with explicit boundaries, confidence tags, and failure conditions.
3. Combine distant branches by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build prototypes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 509 - Innovation and Novelty Engine: Loop Wiring Move

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is loop wiring move, which means the model must mine gaps, combine distant branches, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must upgrade the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mine gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Combine distant branches with explicit boundaries, confidence tags, and failure conditions.
3. Build prototypes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test novelty function through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 510 - Innovation and Novelty Engine: Tool Conversion Move

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is tool conversion move, which means the model must combine distant branches, build prototypes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must index the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Combine distant branches before expanding; never allow raw accumulation to masquerade as depth.
2. Build prototypes with explicit boundaries, confidence tags, and failure conditions.
3. Test novelty function by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mine gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 511 - Innovation and Novelty Engine: Retrieval and Metadata Move

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is retrieval and metadata move, which means the model must build prototypes, test novelty function, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must train the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build prototypes before expanding; never allow raw accumulation to masquerade as depth.
2. Test novelty function with explicit boundaries, confidence tags, and failure conditions.
3. Mine gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Combine distant branches through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 512 - Innovation and Novelty Engine: Agent Role Assignment

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is agent role assignment, which means the model must test novelty function, mine gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must extract the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test novelty function before expanding; never allow raw accumulation to masquerade as depth.
2. Mine gaps with explicit boundaries, confidence tags, and failure conditions.
3. Combine distant branches by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build prototypes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 513 - Innovation and Novelty Engine: Court of Destruction Pass

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is court of destruction pass, which means the model must mine gaps, combine distant branches, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must compress the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mine gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Combine distant branches with explicit boundaries, confidence tags, and failure conditions.
3. Build prototypes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test novelty function through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 514 - Innovation and Novelty Engine: Repair and Reconstruction Pass

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is repair and reconstruction pass, which means the model must combine distant branches, build prototypes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must map the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Combine distant branches before expanding; never allow raw accumulation to masquerade as depth.
2. Build prototypes with explicit boundaries, confidence tags, and failure conditions.
3. Test novelty function by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mine gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 515 - Innovation and Novelty Engine: Training and Mastery Drill

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is training and mastery drill, which means the model must build prototypes, test novelty function, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must route the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build prototypes before expanding; never allow raw accumulation to masquerade as depth.
2. Test novelty function with explicit boundaries, confidence tags, and failure conditions.
3. Mine gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Combine distant branches through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 516 - Innovation and Novelty Engine: Output Interface Contract

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is output interface contract, which means the model must test novelty function, mine gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must score the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test novelty function before expanding; never allow raw accumulation to masquerade as depth.
2. Mine gaps with explicit boundaries, confidence tags, and failure conditions.
3. Combine distant branches by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build prototypes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 517 - Innovation and Novelty Engine: Failure Mode Catalogue

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is failure mode catalogue, which means the model must mine gaps, combine distant branches, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must separate the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Mine gaps before expanding; never allow raw accumulation to masquerade as depth.
2. Combine distant branches with explicit boundaries, confidence tags, and failure conditions.
3. Build prototypes by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Test novelty function through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 518 - Innovation and Novelty Engine: Update and Versioning Rule

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is update and versioning rule, which means the model must combine distant branches, build prototypes, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must convert the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Combine distant branches before expanding; never allow raw accumulation to masquerade as depth.
2. Build prototypes with explicit boundaries, confidence tags, and failure conditions.
3. Test novelty function by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Mine gaps through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 519 - Innovation and Novelty Engine: Novelty and Hypothesis Gate

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is novelty and hypothesis gate, which means the model must build prototypes, test novelty function, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must test the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Build prototypes before expanding; never allow raw accumulation to masquerade as depth.
2. Test novelty function with explicit boundaries, confidence tags, and failure conditions.
3. Mine gaps by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Combine distant branches through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 520 - Innovation and Novelty Engine: Final Prompt Clause

This page belongs to Part 26: Innovation and Novelty Engine. Its task is to extracts gaps, unexplored combinations, possible inventions, measures, methods, and research programs. The local module is final prompt clause, which means the model must test novelty function, mine gaps, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must repair the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Test novelty function before expanding; never allow raw accumulation to masquerade as depth.
2. Mine gaps with explicit boundaries, confidence tags, and failure conditions.
3. Combine distant branches by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Build prototypes through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 521 - Safety, Risk, and Governance Layer: Definition of the Operating Unit

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is definition of the operating unit, which means the model must classify risk, limit overgeneralization, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify risk before expanding; never allow raw accumulation to masquerade as depth.
2. Limit overgeneralization with explicit boundaries, confidence tags, and failure conditions.
3. Write safe-use rules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 522 - Safety, Risk, and Governance Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is scope boundary and non-scope guard, which means the model must limit overgeneralization, write safe-use rules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Limit overgeneralization before expanding; never allow raw accumulation to masquerade as depth.
2. Write safe-use rules with explicit boundaries, confidence tags, and failure conditions.
3. Log uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 523 - Safety, Risk, and Governance Layer: Kernel Extraction Move

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is kernel extraction move, which means the model must write safe-use rules, log uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write safe-use rules before expanding; never allow raw accumulation to masquerade as depth.
2. Log uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Classify risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Limit overgeneralization through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 524 - Safety, Risk, and Governance Layer: Difference-Making Test

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is difference-making test, which means the model must log uncertainty, classify risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Classify risk with explicit boundaries, confidence tags, and failure conditions.
3. Limit overgeneralization by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write safe-use rules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 525 - Safety, Risk, and Governance Layer: Confidence and Evidence Tagging

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is confidence and evidence tagging, which means the model must classify risk, limit overgeneralization, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify risk before expanding; never allow raw accumulation to masquerade as depth.
2. Limit overgeneralization with explicit boundaries, confidence tags, and failure conditions.
3. Write safe-use rules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 526 - Safety, Risk, and Governance Layer: Concept Boundary Split

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is concept boundary split, which means the model must limit overgeneralization, write safe-use rules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Limit overgeneralization before expanding; never allow raw accumulation to masquerade as depth.
2. Write safe-use rules with explicit boundaries, confidence tags, and failure conditions.
3. Log uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 527 - Safety, Risk, and Governance Layer: Problem Deepening Method

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is problem deepening method, which means the model must write safe-use rules, log uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write safe-use rules before expanding; never allow raw accumulation to masquerade as depth.
2. Log uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Classify risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Limit overgeneralization through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 528 - Safety, Risk, and Governance Layer: Skill Conversion Move

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is skill conversion move, which means the model must log uncertainty, classify risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Classify risk with explicit boundaries, confidence tags, and failure conditions.
3. Limit overgeneralization by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write safe-use rules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 529 - Safety, Risk, and Governance Layer: Loop Wiring Move

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is loop wiring move, which means the model must classify risk, limit overgeneralization, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must index the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify risk before expanding; never allow raw accumulation to masquerade as depth.
2. Limit overgeneralization with explicit boundaries, confidence tags, and failure conditions.
3. Write safe-use rules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 530 - Safety, Risk, and Governance Layer: Tool Conversion Move

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is tool conversion move, which means the model must limit overgeneralization, write safe-use rules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must train the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Limit overgeneralization before expanding; never allow raw accumulation to masquerade as depth.
2. Write safe-use rules with explicit boundaries, confidence tags, and failure conditions.
3. Log uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 531 - Safety, Risk, and Governance Layer: Retrieval and Metadata Move

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is retrieval and metadata move, which means the model must write safe-use rules, log uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must extract the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write safe-use rules before expanding; never allow raw accumulation to masquerade as depth.
2. Log uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Classify risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Limit overgeneralization through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 532 - Safety, Risk, and Governance Layer: Agent Role Assignment

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is agent role assignment, which means the model must log uncertainty, classify risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must compress the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Classify risk with explicit boundaries, confidence tags, and failure conditions.
3. Limit overgeneralization by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write safe-use rules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 533 - Safety, Risk, and Governance Layer: Court of Destruction Pass

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is court of destruction pass, which means the model must classify risk, limit overgeneralization, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must map the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify risk before expanding; never allow raw accumulation to masquerade as depth.
2. Limit overgeneralization with explicit boundaries, confidence tags, and failure conditions.
3. Write safe-use rules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 534 - Safety, Risk, and Governance Layer: Repair and Reconstruction Pass

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is repair and reconstruction pass, which means the model must limit overgeneralization, write safe-use rules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must route the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Limit overgeneralization before expanding; never allow raw accumulation to masquerade as depth.
2. Write safe-use rules with explicit boundaries, confidence tags, and failure conditions.
3. Log uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 535 - Safety, Risk, and Governance Layer: Training and Mastery Drill

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is training and mastery drill, which means the model must write safe-use rules, log uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must score the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write safe-use rules before expanding; never allow raw accumulation to masquerade as depth.
2. Log uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Classify risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Limit overgeneralization through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 536 - Safety, Risk, and Governance Layer: Output Interface Contract

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is output interface contract, which means the model must log uncertainty, classify risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must separate the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Classify risk with explicit boundaries, confidence tags, and failure conditions.
3. Limit overgeneralization by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write safe-use rules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 537 - Safety, Risk, and Governance Layer: Failure Mode Catalogue

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is failure mode catalogue, which means the model must classify risk, limit overgeneralization, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must convert the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Classify risk before expanding; never allow raw accumulation to masquerade as depth.
2. Limit overgeneralization with explicit boundaries, confidence tags, and failure conditions.
3. Write safe-use rules by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Log uncertainty through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 538 - Safety, Risk, and Governance Layer: Update and Versioning Rule

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is update and versioning rule, which means the model must limit overgeneralization, write safe-use rules, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must test the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Limit overgeneralization before expanding; never allow raw accumulation to masquerade as depth.
2. Write safe-use rules with explicit boundaries, confidence tags, and failure conditions.
3. Log uncertainty by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Classify risk through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 539 - Safety, Risk, and Governance Layer: Novelty and Hypothesis Gate

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is novelty and hypothesis gate, which means the model must write safe-use rules, log uncertainty, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must repair the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write safe-use rules before expanding; never allow raw accumulation to masquerade as depth.
2. Log uncertainty with explicit boundaries, confidence tags, and failure conditions.
3. Classify risk by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Limit overgeneralization through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 540 - Safety, Risk, and Governance Layer: Final Prompt Clause

This page belongs to Part 27: Safety, Risk, and Governance Layer. Its task is to prevents misuse, overclaiming, unsafe generalization, epistemic harm, and context collapse. The local module is final prompt clause, which means the model must log uncertainty, classify risk, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must upgrade the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Log uncertainty before expanding; never allow raw accumulation to masquerade as depth.
2. Classify risk with explicit boundaries, confidence tags, and failure conditions.
3. Limit overgeneralization by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write safe-use rules through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 541 - Memory, Versioning, and Update Layer: Definition of the Operating Unit

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is definition of the operating unit, which means the model must version the base, record deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Version the base before expanding; never allow raw accumulation to masquerade as depth.
2. Record deltas with explicit boundaries, confidence tags, and failure conditions.
3. Schedule reviews by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rebuild changed sections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 542 - Memory, Versioning, and Update Layer: Scope Boundary and Non-Scope Guard

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is scope boundary and non-scope guard, which means the model must record deltas, schedule reviews, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Schedule reviews with explicit boundaries, confidence tags, and failure conditions.
3. Rebuild changed sections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Version the base through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 543 - Memory, Versioning, and Update Layer: Kernel Extraction Move

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is kernel extraction move, which means the model must schedule reviews, rebuild changed sections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Schedule reviews before expanding; never allow raw accumulation to masquerade as depth.
2. Rebuild changed sections with explicit boundaries, confidence tags, and failure conditions.
3. Version the base by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 544 - Memory, Versioning, and Update Layer: Difference-Making Test

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is difference-making test, which means the model must rebuild changed sections, version the base, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rebuild changed sections before expanding; never allow raw accumulation to masquerade as depth.
2. Version the base with explicit boundaries, confidence tags, and failure conditions.
3. Record deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Schedule reviews through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 545 - Memory, Versioning, and Update Layer: Confidence and Evidence Tagging

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is confidence and evidence tagging, which means the model must version the base, record deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Version the base before expanding; never allow raw accumulation to masquerade as depth.
2. Record deltas with explicit boundaries, confidence tags, and failure conditions.
3. Schedule reviews by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rebuild changed sections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 546 - Memory, Versioning, and Update Layer: Concept Boundary Split

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is concept boundary split, which means the model must record deltas, schedule reviews, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Schedule reviews with explicit boundaries, confidence tags, and failure conditions.
3. Rebuild changed sections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Version the base through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 547 - Memory, Versioning, and Update Layer: Problem Deepening Method

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is problem deepening method, which means the model must schedule reviews, rebuild changed sections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Schedule reviews before expanding; never allow raw accumulation to masquerade as depth.
2. Rebuild changed sections with explicit boundaries, confidence tags, and failure conditions.
3. Version the base by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 548 - Memory, Versioning, and Update Layer: Skill Conversion Move

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is skill conversion move, which means the model must rebuild changed sections, version the base, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rebuild changed sections before expanding; never allow raw accumulation to masquerade as depth.
2. Version the base with explicit boundaries, confidence tags, and failure conditions.
3. Record deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Schedule reviews through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 549 - Memory, Versioning, and Update Layer: Loop Wiring Move

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is loop wiring move, which means the model must version the base, record deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must train the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Version the base before expanding; never allow raw accumulation to masquerade as depth.
2. Record deltas with explicit boundaries, confidence tags, and failure conditions.
3. Schedule reviews by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rebuild changed sections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 550 - Memory, Versioning, and Update Layer: Tool Conversion Move

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is tool conversion move, which means the model must record deltas, schedule reviews, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must extract the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Schedule reviews with explicit boundaries, confidence tags, and failure conditions.
3. Rebuild changed sections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Version the base through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 551 - Memory, Versioning, and Update Layer: Retrieval and Metadata Move

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is retrieval and metadata move, which means the model must schedule reviews, rebuild changed sections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must compress the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Schedule reviews before expanding; never allow raw accumulation to masquerade as depth.
2. Rebuild changed sections with explicit boundaries, confidence tags, and failure conditions.
3. Version the base by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 552 - Memory, Versioning, and Update Layer: Agent Role Assignment

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is agent role assignment, which means the model must rebuild changed sections, version the base, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must map the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rebuild changed sections before expanding; never allow raw accumulation to masquerade as depth.
2. Version the base with explicit boundaries, confidence tags, and failure conditions.
3. Record deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Schedule reviews through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 553 - Memory, Versioning, and Update Layer: Court of Destruction Pass

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is court of destruction pass, which means the model must version the base, record deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must route the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Version the base before expanding; never allow raw accumulation to masquerade as depth.
2. Record deltas with explicit boundaries, confidence tags, and failure conditions.
3. Schedule reviews by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rebuild changed sections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 554 - Memory, Versioning, and Update Layer: Repair and Reconstruction Pass

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is repair and reconstruction pass, which means the model must record deltas, schedule reviews, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must score the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Schedule reviews with explicit boundaries, confidence tags, and failure conditions.
3. Rebuild changed sections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Version the base through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 555 - Memory, Versioning, and Update Layer: Training and Mastery Drill

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is training and mastery drill, which means the model must schedule reviews, rebuild changed sections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must separate the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Schedule reviews before expanding; never allow raw accumulation to masquerade as depth.
2. Rebuild changed sections with explicit boundaries, confidence tags, and failure conditions.
3. Version the base by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 556 - Memory, Versioning, and Update Layer: Output Interface Contract

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is output interface contract, which means the model must rebuild changed sections, version the base, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must convert the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rebuild changed sections before expanding; never allow raw accumulation to masquerade as depth.
2. Version the base with explicit boundaries, confidence tags, and failure conditions.
3. Record deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Schedule reviews through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 557 - Memory, Versioning, and Update Layer: Failure Mode Catalogue

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is failure mode catalogue, which means the model must version the base, record deltas, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must test the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Version the base before expanding; never allow raw accumulation to masquerade as depth.
2. Record deltas with explicit boundaries, confidence tags, and failure conditions.
3. Schedule reviews by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Rebuild changed sections through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 558 - Memory, Versioning, and Update Layer: Update and Versioning Rule

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is update and versioning rule, which means the model must record deltas, schedule reviews, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must repair the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Record deltas before expanding; never allow raw accumulation to masquerade as depth.
2. Schedule reviews with explicit boundaries, confidence tags, and failure conditions.
3. Rebuild changed sections by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Version the base through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 559 - Memory, Versioning, and Update Layer: Novelty and Hypothesis Gate

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is novelty and hypothesis gate, which means the model must schedule reviews, rebuild changed sections, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must upgrade the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Schedule reviews before expanding; never allow raw accumulation to masquerade as depth.
2. Rebuild changed sections with explicit boundaries, confidence tags, and failure conditions.
3. Version the base by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Record deltas through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 560 - Memory, Versioning, and Update Layer: Final Prompt Clause

This page belongs to Part 28: Memory, Versioning, and Update Layer. Its task is to makes the base evolve through logs, versions, deltas, review cycles, and reconstruction memory. The local module is final prompt clause, which means the model must rebuild changed sections, version the base, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must index the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Rebuild changed sections before expanding; never allow raw accumulation to masquerade as depth.
2. Version the base with explicit boundaries, confidence tags, and failure conditions.
3. Record deltas by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Schedule reviews through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 561 - Output Templates and Interfaces: Definition of the Operating Unit

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is definition of the operating unit, which means the model must format reusable cards, write reports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Format reusable cards before expanding; never allow raw accumulation to masquerade as depth.
2. Write reports with explicit boundaries, confidence tags, and failure conditions.
3. Create machine records by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Standardize exports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 562 - Output Templates and Interfaces: Scope Boundary and Non-Scope Guard

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is scope boundary and non-scope guard, which means the model must write reports, create machine records, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reports before expanding; never allow raw accumulation to masquerade as depth.
2. Create machine records with explicit boundaries, confidence tags, and failure conditions.
3. Standardize exports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Format reusable cards through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 563 - Output Templates and Interfaces: Kernel Extraction Move

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is kernel extraction move, which means the model must create machine records, standardize exports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create machine records before expanding; never allow raw accumulation to masquerade as depth.
2. Standardize exports with explicit boundaries, confidence tags, and failure conditions.
3. Format reusable cards by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 564 - Output Templates and Interfaces: Difference-Making Test

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is difference-making test, which means the model must standardize exports, format reusable cards, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Standardize exports before expanding; never allow raw accumulation to masquerade as depth.
2. Format reusable cards with explicit boundaries, confidence tags, and failure conditions.
3. Write reports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create machine records through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 565 - Output Templates and Interfaces: Confidence and Evidence Tagging

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is confidence and evidence tagging, which means the model must format reusable cards, write reports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Format reusable cards before expanding; never allow raw accumulation to masquerade as depth.
2. Write reports with explicit boundaries, confidence tags, and failure conditions.
3. Create machine records by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Standardize exports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 566 - Output Templates and Interfaces: Concept Boundary Split

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is concept boundary split, which means the model must write reports, create machine records, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reports before expanding; never allow raw accumulation to masquerade as depth.
2. Create machine records with explicit boundaries, confidence tags, and failure conditions.
3. Standardize exports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Format reusable cards through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 567 - Output Templates and Interfaces: Problem Deepening Method

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is problem deepening method, which means the model must create machine records, standardize exports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create machine records before expanding; never allow raw accumulation to masquerade as depth.
2. Standardize exports with explicit boundaries, confidence tags, and failure conditions.
3. Format reusable cards by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 568 - Output Templates and Interfaces: Skill Conversion Move

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is skill conversion move, which means the model must standardize exports, format reusable cards, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Standardize exports before expanding; never allow raw accumulation to masquerade as depth.
2. Format reusable cards with explicit boundaries, confidence tags, and failure conditions.
3. Write reports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create machine records through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 569 - Output Templates and Interfaces: Loop Wiring Move

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is loop wiring move, which means the model must format reusable cards, write reports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must extract the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Format reusable cards before expanding; never allow raw accumulation to masquerade as depth.
2. Write reports with explicit boundaries, confidence tags, and failure conditions.
3. Create machine records by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Standardize exports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 570 - Output Templates and Interfaces: Tool Conversion Move

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is tool conversion move, which means the model must write reports, create machine records, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must compress the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reports before expanding; never allow raw accumulation to masquerade as depth.
2. Create machine records with explicit boundaries, confidence tags, and failure conditions.
3. Standardize exports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Format reusable cards through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 571 - Output Templates and Interfaces: Retrieval and Metadata Move

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is retrieval and metadata move, which means the model must create machine records, standardize exports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must map the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create machine records before expanding; never allow raw accumulation to masquerade as depth.
2. Standardize exports with explicit boundaries, confidence tags, and failure conditions.
3. Format reusable cards by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 572 - Output Templates and Interfaces: Agent Role Assignment

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is agent role assignment, which means the model must standardize exports, format reusable cards, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must route the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Standardize exports before expanding; never allow raw accumulation to masquerade as depth.
2. Format reusable cards with explicit boundaries, confidence tags, and failure conditions.
3. Write reports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create machine records through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 573 - Output Templates and Interfaces: Court of Destruction Pass

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is court of destruction pass, which means the model must format reusable cards, write reports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must score the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Format reusable cards before expanding; never allow raw accumulation to masquerade as depth.
2. Write reports with explicit boundaries, confidence tags, and failure conditions.
3. Create machine records by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Standardize exports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 574 - Output Templates and Interfaces: Repair and Reconstruction Pass

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is repair and reconstruction pass, which means the model must write reports, create machine records, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must separate the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reports before expanding; never allow raw accumulation to masquerade as depth.
2. Create machine records with explicit boundaries, confidence tags, and failure conditions.
3. Standardize exports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Format reusable cards through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 575 - Output Templates and Interfaces: Training and Mastery Drill

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is training and mastery drill, which means the model must create machine records, standardize exports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must convert the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create machine records before expanding; never allow raw accumulation to masquerade as depth.
2. Standardize exports with explicit boundaries, confidence tags, and failure conditions.
3. Format reusable cards by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 576 - Output Templates and Interfaces: Output Interface Contract

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is output interface contract, which means the model must standardize exports, format reusable cards, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must test the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Standardize exports before expanding; never allow raw accumulation to masquerade as depth.
2. Format reusable cards with explicit boundaries, confidence tags, and failure conditions.
3. Write reports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create machine records through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 577 - Output Templates and Interfaces: Failure Mode Catalogue

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is failure mode catalogue, which means the model must format reusable cards, write reports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The model must spend deep reasoning on knots, contradictions, boundary cases, and high-impact differences, not on ornamental prose. In this module, the assistant must not merely explain the field; it must repair the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Format reusable cards before expanding; never allow raw accumulation to masquerade as depth.
2. Write reports with explicit boundaries, confidence tags, and failure conditions.
3. Create machine records by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Standardize exports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 578 - Output Templates and Interfaces: Update and Versioning Rule

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is update and versioning rule, which means the model must write reports, create machine records, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

The final system must be ready for human reading, RAG chunking, GraphRAG relation mapping, and agentic execution. In this module, the assistant must not merely explain the field; it must upgrade the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Write reports before expanding; never allow raw accumulation to masquerade as depth.
2. Create machine records with explicit boundaries, confidence tags, and failure conditions.
3. Standardize exports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Format reusable cards through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 579 - Output Templates and Interfaces: Novelty and Hypothesis Gate

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is novelty and hypothesis gate, which means the model must create machine records, standardize exports, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A beautiful term without function is treated as noise until it becomes a tool, distinction, metric, or repeatable move. In this module, the assistant must not merely explain the field; it must index the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Create machine records before expanding; never allow raw accumulation to masquerade as depth.
2. Standardize exports with explicit boundaries, confidence tags, and failure conditions.
3. Format reusable cards by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Write reports through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 580 - Output Templates and Interfaces: Final Prompt Clause

This page belongs to Part 29: Output Templates and Interfaces. Its task is to provides ready formats for articles, OS cards, RAG records, agent briefs, skill cards, and court reports. The local module is final prompt clause, which means the model must standardize exports, format reusable cards, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Compression is not deletion. Compression preserves the generative core that can rebuild the branches of the field. In this module, the assistant must not merely explain the field; it must train the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Standardize exports before expanding; never allow raw accumulation to masquerade as depth.
2. Format reusable cards with explicit boundaries, confidence tags, and failure conditions.
3. Write reports by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Create machine records through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 581 - Master Execution Protocol: Definition of the Operating Unit

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is definition of the operating unit, which means the model must assemble all layers, run final courts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble all layers before expanding; never allow raw accumulation to masquerade as depth.
2. Run final courts with explicit boundaries, confidence tags, and failure conditions.
3. Emit the os by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Include a copy-ready compact card through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat definition of the operating unit as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 582 - Master Execution Protocol: Scope Boundary and Non-Scope Guard

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is scope boundary and non-scope guard, which means the model must run final courts, emit the OS, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run final courts before expanding; never allow raw accumulation to masquerade as depth.
2. Emit the os with explicit boundaries, confidence tags, and failure conditions.
3. Include a copy-ready compact card by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble all layers through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat scope boundary and non-scope guard as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 583 - Master Execution Protocol: Kernel Extraction Move

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is kernel extraction move, which means the model must emit the OS, include a copy-ready compact card, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Emit the os before expanding; never allow raw accumulation to masquerade as depth.
2. Include a copy-ready compact card with explicit boundaries, confidence tags, and failure conditions.
3. Assemble all layers by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run final courts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat kernel extraction move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 584 - Master Execution Protocol: Difference-Making Test

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is difference-making test, which means the model must include a copy-ready compact card, assemble all layers, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Include a copy-ready compact card before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble all layers with explicit boundaries, confidence tags, and failure conditions.
3. Run final courts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Emit the os through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat difference-making test as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 585 - Master Execution Protocol: Confidence and Evidence Tagging

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is confidence and evidence tagging, which means the model must assemble all layers, run final courts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble all layers before expanding; never allow raw accumulation to masquerade as depth.
2. Run final courts with explicit boundaries, confidence tags, and failure conditions.
3. Emit the os by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Include a copy-ready compact card through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat confidence and evidence tagging as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 586 - Master Execution Protocol: Concept Boundary Split

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is concept boundary split, which means the model must run final courts, emit the OS, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run final courts before expanding; never allow raw accumulation to masquerade as depth.
2. Emit the os with explicit boundaries, confidence tags, and failure conditions.
3. Include a copy-ready compact card by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble all layers through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat concept boundary split as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 587 - Master Execution Protocol: Problem Deepening Method

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is problem deepening method, which means the model must emit the OS, include a copy-ready compact card, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Emit the os before expanding; never allow raw accumulation to masquerade as depth.
2. Include a copy-ready compact card with explicit boundaries, confidence tags, and failure conditions.
3. Assemble all layers by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run final courts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat problem deepening method as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 588 - Master Execution Protocol: Skill Conversion Move

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is skill conversion move, which means the model must include a copy-ready compact card, assemble all layers, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Include a copy-ready compact card before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble all layers with explicit boundaries, confidence tags, and failure conditions.
3. Run final courts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Emit the os through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat skill conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 589 - Master Execution Protocol: Loop Wiring Move

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is loop wiring move, which means the model must assemble all layers, run final courts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must compress the relevant tools into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble all layers before expanding; never allow raw accumulation to masquerade as depth.
2. Run final courts with explicit boundaries, confidence tags, and failure conditions.
3. Emit the os by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Include a copy-ready compact card through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat loop wiring move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 590 - Master Execution Protocol: Tool Conversion Move

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is tool conversion move, which means the model must run final courts, emit the OS, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must map the relevant sources into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run final courts before expanding; never allow raw accumulation to masquerade as depth.
2. Emit the os with explicit boundaries, confidence tags, and failure conditions.
3. Include a copy-ready compact card by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble all layers through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat tool conversion move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 591 - Master Execution Protocol: Retrieval and Metadata Move

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is retrieval and metadata move, which means the model must emit the OS, include a copy-ready compact card, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must route the relevant assumptions into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Emit the os before expanding; never allow raw accumulation to masquerade as depth.
2. Include a copy-ready compact card with explicit boundaries, confidence tags, and failure conditions.
3. Assemble all layers by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run final courts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat retrieval and metadata move as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 592 - Master Execution Protocol: Agent Role Assignment

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is agent role assignment, which means the model must include a copy-ready compact card, assemble all layers, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must score the relevant outputs into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Include a copy-ready compact card before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble all layers with explicit boundaries, confidence tags, and failure conditions.
3. Run final courts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Emit the os through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat agent role assignment as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 593 - Master Execution Protocol: Court of Destruction Pass

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is court of destruction pass, which means the model must assemble all layers, run final courts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must separate the relevant interfaces into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble all layers before expanding; never allow raw accumulation to masquerade as depth.
2. Run final courts with explicit boundaries, confidence tags, and failure conditions.
3. Emit the os by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Include a copy-ready compact card through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat court of destruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 594 - Master Execution Protocol: Repair and Reconstruction Pass

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is repair and reconstruction pass, which means the model must run final courts, emit the OS, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must convert the relevant risks into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run final courts before expanding; never allow raw accumulation to masquerade as depth.
2. Emit the os with explicit boundaries, confidence tags, and failure conditions.
3. Include a copy-ready compact card by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble all layers through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat repair and reconstruction pass as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 595 - Master Execution Protocol: Training and Mastery Drill

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is training and mastery drill, which means the model must emit the OS, include a copy-ready compact card, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must test the relevant concepts into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Emit the os before expanding; never allow raw accumulation to masquerade as depth.
2. Include a copy-ready compact card with explicit boundaries, confidence tags, and failure conditions.
3. Assemble all layers by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run final courts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat training and mastery drill as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 596 - Master Execution Protocol: Output Interface Contract

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is output interface contract, which means the model must include a copy-ready compact card, assemble all layers, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must repair the relevant claims into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Include a copy-ready compact card before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble all layers with explicit boundaries, confidence tags, and failure conditions.
3. Run final courts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Emit the os through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat output interface contract as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 597 - Master Execution Protocol: Failure Mode Catalogue

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is failure mode catalogue, which means the model must assemble all layers, run final courts, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every strong claim must carry a confidence tag; every weak claim must be protected from becoming false certainty. In this module, the assistant must not merely explain the field; it must upgrade the relevant problems into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Assemble all layers before expanding; never allow raw accumulation to masquerade as depth.
2. Run final courts with explicit boundaries, confidence tags, and failure conditions.
3. Emit the os by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Include a copy-ready compact card through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat failure mode catalogue as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 598 - Master Execution Protocol: Update and Versioning Rule

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is update and versioning rule, which means the model must run final courts, emit the OS, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

A HyperKB is not a library. It is a cognitive operating system with kernel, layers, loops, courts, and interfaces. In this module, the assistant must not merely explain the field; it must index the relevant relations into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Run final courts before expanding; never allow raw accumulation to masquerade as depth.
2. Emit the os with explicit boundaries, confidence tags, and failure conditions.
3. Include a copy-ready compact card by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Assemble all layers through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat update and versioning rule as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 599 - Master Execution Protocol: Novelty and Hypothesis Gate

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is novelty and hypothesis gate, which means the model must emit the OS, include a copy-ready compact card, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Every section must be usable by at least one actor: researcher, learner, teacher, agent, writer, evaluator, builder, or updater. In this module, the assistant must not merely explain the field; it must train the relevant skills into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Emit the os before expanding; never allow raw accumulation to masquerade as depth.
2. Include a copy-ready compact card with explicit boundaries, confidence tags, and failure conditions.
3. Assemble all layers by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Run final courts through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat novelty and hypothesis gate as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.

Page 600 - Master Execution Protocol: Final Prompt Clause

This page belongs to Part 30: Master Execution Protocol. Its task is to assembles all layers into one copy-ready production engine for any field or topic. The local module is final prompt clause, which means the model must include a copy-ready compact card, assemble all layers, and keep the result usable by humans, RAG systems, GraphRAG systems, and agentic workflows.

Operating Doctrine

Knowledge is accepted only when it changes understanding, decision, generation, critique, training, retrieval, or update. In this module, the assistant must not merely explain the field; it must extract the relevant loops into a unit that can be stored, retrieved, attacked, trained, and updated. The unit is accepted only if it has a visible role in the HyperKB architecture: kernel, concept layer, evidence layer, problem layer, method layer, skill layer, loop layer, tool layer, retrieval layer, agent layer, court layer, or evaluation layer.

Construction Moves

1. Include a copy-ready compact card before expanding; never allow raw accumulation to masquerade as depth.
2. Assemble all layers with explicit boundaries, confidence tags, and failure conditions.
3. Run final courts by translating the result into a prompt clause, a skill card, a loop step, a tool field, or a court question.
4. Emit the os through reconstruction memory so that every improvement leaves a trace.

MCD and QMCD Filter

Apply the difference-making test: what changes if this unit is included, removed, weakened, or misunderstood? If it changes nothing, compress or delete it. If it changes a decision, a distinction, a method, a risk, a training path, or a retrieval route, rank it as operational. Use QMCD only as disciplined orientation: centrality, confidence, maturity, risk, and transferability must be tagged without pretending to mathematical certainty when the data does not support it.

HyperSkill Extraction

Turn this module into a trainable capability. Name the micro-skill, define the observable behavior, state the novice error, write one drill, and specify the mastery sign. A skill is not accepted until a user or agent can practice it. A HyperSkill appears when several micro-skills cooperate: extraction, distinction, evidence control, relation mapping, critique, repair, and reuse.

HyperLoop Wiring

Connect the module to a recurring loop. The minimum loop is: ingest -> distinguish -> tag -> operate -> destroy -> repair -> store -> reuse. The loop must have a trigger, an input, an output, a failure signal, and an update rule. A loop that does not change the next run is only a ritual and must be rewritten.

Court of Destruction

Attack this module with these questions: Is the claim unsupported? Is the concept merely renamed? Is the field reduced too aggressively? Is the output too broad to execute? Is confidence higher than evidence? Is the tool usable? Is the skill trainable? Is the loop self-improving? If any answer exposes weakness, repair the module before continuing.

Copy-Ready Prompt Clause

When building the HyperKB for the chosen topic, treat final prompt clause as a mandatory OS component. For every relevant item, state its function, boundary, evidence status, MCD value, operational use, failure mode, and update path. Do not deliver ornamental explanation; deliver a working knowledge unit that can survive retrieval, agent execution, and hostile review.