MATI: Moussaif Agentic Theory Internalization

A Fine-Tuning Theory and Toolchain for Internalizing Mathematical-Agentic Operating Systems into Large Language Models

AGILLM-FTOS v1.0 | Moussaif Zouhair | 12 August 2026

Abstract. This paper proposes MATI - Moussaif Agentic Theory Internalization - and its engineering realization, AGILLM-FTOS. The central hypothesis is that fine-tuning a large language model on an agentic operating system must transfer executable reasoning structure, not merely vocabulary or factual content. MATI therefore represents a theory as a typed graph of axioms, equations, questions, evidence contracts, tools, skills, bounded loops, authorization gates, falsifiers, world-effect tests, and memory transitions. Training data are compiled from this graph into supervised examples, preference pairs, process-verifier traces, tool trajectories, and verifiable reinforcement-learning tasks. The target model is evaluated by whether it can preserve mathematical relations under perturbation, ask high-value governed questions, choose sparse reasoning paths, invoke tools appropriately, respect reversible authorization, verify external effects, and reconstruct memory after execution.

The proposed framework is explicitly a research program, not a claim of achieved AGI. Its contribution is a falsifiable bridge between a large external knowledge-and-agent architecture and post-training methods that can internalize parts of that architecture into model behavior. It retains external HyperKB memory where provenance and updateability matter, while selectively parameterizing recurring control laws. The result is a hybrid model-plus-OS design in which the model learns how to operate the OS rather than attempting to memorize the entire OS.

Contributions: (1) an agentic state-space formalization; (2) a multi-term MATI training objective; (3) a TheoryGraph compiler; (4) curricula for MCD/QMCD question governance, HyperSkills, HyperLoops, tools, memory, and courts; (5) a world-effect and reversibility reward model; and (6) an evaluation suite centered on theory internalization, routing regret, evidence calibration, loop closure, and verified state change.

Operationalization. For title, abstract, and contributions, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for representation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target semantic invariance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out graph perturbation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

2. Problem Formulation: From Knowledge Fine-Tuning to Operating-System Internalization

Conventional fine-tuning is often framed as conditional distribution adaptation: given input x, adjust parameters theta so that p_theta(y|x) better matches a target distribution. That abstraction is incomplete for an agentic operating system. The target object contains policies that modify future context, call tools, allocate compute, consult external knowledge, branch into candidates, enforce permissions, and write memory. A response can be linguistically correct while being operationally wrong because it asked the wrong question, used the wrong evidence grade, skipped a gate, or failed to verify the world state after acting.

MATI therefore treats the training target as a transition system. A sample is not only (prompt, answer); it is a trajectory with latent and observable control variables. The model should learn which parts belong in parameters, which must be retrieved from HKOS, which require a tool, and which must remain blocked until authorization. This distinction is essential for large mathematical theory collections because equations, definitions, assumptions, and counterexamples have different epistemic roles. Memorizing a formula without its domain of validity produces symbolic mimicry rather than theory competence.

The design principle is selective internalization: parameterize stable control laws and frequently reused abstractions; retrieve mutable facts and large bodies of evidence; execute exact computation in tools; retain provenance externally; and use verification to promote only reproducible state changes. Fine-tuning becomes a compiler from an external cognitive architecture into a hybrid parametric/non-parametric policy.

Operationalization. For problem formulation: from knowledge fine-tuning to operating-system internalization, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for optimization because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target credit assignment: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out ablation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

3. State of the Art and Research Gap

Parameter-efficient methods such as LoRA [1] and QLoRA [2] make adaptation of large models feasible without updating every base parameter. Preference methods such as DPO [3] simplify alignment against ranked outputs. Reasoning-oriented work increasingly uses reinforcement learning and verifiable rewards; DeepSeek-R1 [9] demonstrates a multi-stage route combining cold-start data with RL, while the Qwen3 technical report [10] describes unified thinking/non-thinking behavior and explicit thinking budgets. These methods motivate MATI's separation between basic theory imitation, reasoning-policy optimization, and adaptive compute allocation.

Agent research provides complementary pieces. ReAct [4] interleaves reasoning and acting; Toolformer [5] trains API-use decisions; Reflexion [6] uses episodic linguistic feedback; RAG [7] couples generation to non-parametric memory. More recent work directly trains agent trajectories: ReTool [11] uses outcome-driven RL for strategic code-tool use, and Agent Lightning [12] formalizes arbitrary agent execution as RL-compatible transitions with credit assignment. Process-verifier work [8] suggests that progress-sensitive step rewards can improve search and RL efficiency, while RLVR research [13] focuses attention on verifiable outcome structure.

The gap addressed here is integration. Existing techniques usually optimize one dimension - adaptation, preferences, tools, retrieval, reasoning, or agents. MATI asks how to compile an entire named mathematical-agentic architecture into a coordinated training distribution with explicit invariants, provenance, permissions, and falsification. Its novelty is therefore architectural composition plus formal evaluation of whether the learned policy preserves the operating semantics of the source theory stack.

Operationalization. For state of the art and research gap, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for dataset engineering because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target provenance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out split isolation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

4. Canonical MOUSSAIF Theory Stack as Training Substrate

Page thesis. Maps MCD, QMCD, HAKIM-MCD, HKOS, HyperSkills, HyperAbilities, HyperLoops, sparse routing, courts, world-effect verification, and self-reconstruction into trainable targets. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

The canonical stack begins with difference extraction and the MCD score, MCD = lambda1*D + lambda2*I + lambda3*T + lambda4*A + lambda5*P + lambda6*C - lambda7*N. QMCD multiplies this score by a question-governance term Omega(Q), turning questions into compute-allocation operators rather than conversational ornaments. HAKIM-MCD then organizes execution as: difference extraction -> MCD scoring -> vascular compute routing -> candidate paths -> falsification court -> tool/GraphRAG -> agency selection -> world-effect loop -> self-reconstruction memory. MATI converts each arrow into explicit training transitions and each invariant into evaluation constraints.

The rest of the OS contributes different target types. HKOS and HyperKB define external knowledge memory; HyperSkills define reusable bounded procedures; HyperAbilities compose skills and reasoning into higher-order competencies; HyperLoops define monitored iteration; Sparse HyperRouting chooses minimal sufficient compute; the Destruction/Demolition courts provide adversarial verification; constitutional T0-T4 gates constrain authority; World-Effect loops demand observable state change; and Self-Reconstruction Memory writes only verified deltas. These are heterogeneous objects and must not be flattened into undifferentiated instruction-response text.

Operationalization. For canonical moussaif theory stack as training substrate, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for agent control because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target bounded execution: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out state verification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

5. MATI State Space and Agentic Markov Semantics

Page thesis. Formalizes the internal agent state S_t=(X,K,Q,E,R,A,M,G) and shows how agentic learning optimizes transitions rather than isolated responses. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

We define the agentic state as S_t = (X_t, K_t, Q_t, E_t, R_t, A_t, M_t, G_t). X is the task/context; K is retrieved and parametric knowledge; Q is the active governed-question set; E is evidence with grades and provenance; R is the routing state over models/tools/skills; A is the authorized action set; M is working and persistent memory; G is governance constraints. An action u_t can be a token sequence, a question, a retrieval, a tool invocation, a skill activation, a loop transition, an authorization request, or a memory write. The environment returns observation o_t and possibly a verifiable state delta.

The policy pi_theta(u_t|S_t) is therefore trained over typed actions. A trajectory tau contains not only text but the sequence of state updates that made the text justified. This allows the same mathematical theory to generate many training regimes: pure theorem explanation, retrieval-conditioned proof, calculator-assisted derivation, disagreement arbitration between agents, or gated real-world execution. The state-space view also exposes where credit assignment belongs: a final success may depend on an earlier question, evidence fetch, or permission check.

Operationalization. For mati state space and agentic markov semantics, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for evaluation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target calibration: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out falsification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

6. Unified MATI Objective Function

Page thesis. Defines a composite loss over semantic fidelity, governed questioning, difference extraction, routing, evidence, tools, loops, world effects, memory, constitutional safety, sparsity, and calibration. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

The composite objective is L_MATI = a_s*L_sem + a_q*L_QMCD + a_d*L_delta + a_r*L_route + a_e*L_evidence + a_t*L_tool + a_h*L_hyper + a_w*L_world + a_m*L_memory + a_c*L_const + a_p*L_process + a_k*L_calibration. L_sem preserves definitions and conceptual relations; L_QMCD ranks and generates high-value questions; L_delta trains difference extraction; L_route trains sparse allocation across tools/models/skills; L_evidence penalizes unsupported claims and misgraded sources; L_tool supervises invocation and argument correctness; L_hyper covers HyperSkill/HyperLoop transitions; L_world rewards verified external progress; L_memory constrains read/write behavior; L_const enforces authorization and reversibility; L_process trains step quality; L_calibration aligns confidence with evidence.

Weights should be curriculum-dependent rather than fixed. Early SFT may emphasize semantic and structural fidelity; later preference optimization can increase question, routing, evidence, and constitutional terms; RL can then focus on process, tools, world effects, and loop closure. A central experimental question is whether joint optimization causes destructive interference. MATI therefore supports adapter decomposition: theory adapter, governance adapter, tool adapter, and agentic-control adapter can be trained separately and then composed or merged after ablation.

Operationalization. For unified mati objective function, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for representation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target semantic invariance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out graph perturbation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

7. TheoryGraph: Compiling Mathematical Theories into a Trainable Graph

Page thesis. Defines nodes for axioms, equations, operators, claims, counterexamples, tools, permissions, and experiments, with typed edges and provenance. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

TheoryGraph is a typed multigraph G_T=(V,E). Node types include Definition, Axiom, Equation, Variable, Assumption, Claim, Counterexample, Question, Evidence, Tool, Skill, Ability, Loop, Permission, Metric, Experiment, FailureMode, and MemoryObject. Edge types include defines, derives, constrains, contradicts, falsifies, requires, invokes, verifies, generalizes, specializes, routes_to, writes_to, and authorized_by. Every edge carries source, confidence, validity scope, timestamp, and canonicality rank.

Training examples are graph programs. A definition node can emit forward recall and inverse-identification tasks; an equation can emit variable interpretation, dimensional checks, derivations, perturbations, missing-term recovery, and counterexamples; a requires edge can become a prerequisite-order task; a falsifies edge becomes an adversarial pair; an invokes edge becomes tool-use supervision. This graph compiler prevents the common failure in large knowledge fine-tuning where repeated prose dominates rare but structurally important relations.

Operationalization. For theorygraph: compiling mathematical theories into a trainable graph, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for optimization because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target credit assignment: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out ablation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

8. Corpus Engineering and Canonicalization

Page thesis. Describes source ranking, deduplication, equation normalization, lineage preservation, contradiction tagging, and train/validation/test isolation. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Corpus construction begins with source registration and cryptographic identity, then semantic deduplication, version lineage, and canonical claim extraction. The data layer preserves RAW versus DERIVED artifacts and prohibits silent overwriting. Equation canonicalization normalizes variable names only in a linked representation, never by mutating the archived source. Contradictions are retained as explicit objects so that the model can learn uncertainty and theory evolution rather than receiving an artificially consistent corpus.

Leakage control is critical. Train, validation, and test partitions should be split by theory subgraph and transformation family, not only by documents, because templated derivations can otherwise leak. A strong held-out set includes equations never seen in the same surface form, compositional tasks requiring two or more theory edges, adversarial distractors, and tasks where the correct action is retrieval or abstention rather than generation. Provenance metadata should survive every transformation into the final dataset row.

Operationalization. For corpus engineering and canonicalization, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for dataset engineering because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target provenance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out split isolation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

9. Mathematical Internalization Curriculum

Page thesis. Transforms formulas and theories into definition, derivation, inverse, perturbation, counterexample, unit-test, and transfer tasks. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

A mathematical curriculum must distinguish symbolic memory from mathematical operation. For each formula MATI generates: definition reconstruction; variable-role identification; domain assumptions; forward calculation; inverse solving; parameter sensitivity; limiting cases; equivalence transformations; counterexample construction; dimensional or type checks; dependency tracing; and transfer to a new scenario. The target is invariant relation preservation under changes of wording, symbol names, and context.

The compiler also generates theorem cards containing Premises, Claim, Derivation Sketch, Falsifier, Tool Check, and Transfer Test. For internally proposed theories such as QMCD, the card explicitly labels empirical status as unvalidated until benchmark evidence exists. This prevents the fine-tuned model from upgrading a research hypothesis into fact. Verifiable mathematical tasks are ideal candidates for RLVR because correctness can often be checked with code, symbolic algebra, or unit tests, but reward design must still penalize invalid reasoning paths that happen to reach the right answer.

Operationalization. For mathematical internalization curriculum, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for agent control because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target bounded execution: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out state verification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

10. Question Governance and QMCD Curriculum

Page thesis. Trains the model to generate and rank questions that maximize expected information and decision value under cost and risk constraints. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Question governance is trained as both generation and ranking. Given S_t, the target question is approximated by Q*_t = argmax_q [Omega(q|S_t) * MCD(q|S_t) - beta*C_compute(q) - gamma*Risk(q)]. Omega encodes expected decision relevance, novelty, uncertainty reduction, and branching impact. Training examples include pairwise preferences between questions, top-k question sets, stop-question decisions, and counterexamples where asking more is wasteful or unsafe. A companion metric, GQG = (ExpectedInformationGain + ExpectedDecisionGain - Cost - Risk) / (1 + QuestionNoise), measures the expected gain from a question relative to cost and noise.

The curriculum should contain missing-information tasks where the correct behavior is to ask, but also over-questioning tasks where the model must proceed with available evidence. This is important because agentic systems can fail through both premature action and endless clarification. MATI calls the optimal boundary question sufficiency: the model should ask until uncertainty about the decision-relevant state is below a task-dependent threshold, then route compute to action or verification.

Operationalization. For question governance and qmcd curriculum, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for evaluation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target calibration: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out falsification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

11. Difference Extraction, Candidate Generation, and Sparse Reasoning

Page thesis. Turns the HAKIM-MCD front half into supervised and preference data for efficient reasoning path selection. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Difference extraction trains the model to represent what changed between observed state, desired state, retrieved evidence, candidate solutions, and prior memory. Instead of rewarding verbosity, the target is a sparse delta vector over task-relevant dimensions. Candidate generation then produces a small set of meaningfully distinct hypotheses or plans. MCD scores each candidate's difference, informativeness, testability, agency value, predictive value, coherence, and noise penalty.

Sparse HyperRouting converts candidate scores into compute allocation. The model learns when a cheap internal answer is sufficient, when retrieval is required, when an exact tool should replace language reasoning, and when disagreement warrants a second model or specialist agent. Routing regret measures the difference between the utility of the chosen route and the best available route under the same budget. Fine-tuning should optimize low regret, not maximal tool use or maximal reasoning length.

Operationalization. For difference extraction, candidate generation, and sparse reasoning, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for representation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target semantic invariance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out graph perturbation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

12. Tool-Use Internalization

Page thesis. Uses structured tool traces, outcome labels, action validity, error recovery, and dynamic interleaving to teach when and how to use external tools. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Tool-use data use a typed trace: intent -> selected tool -> argument schema -> permission check -> call -> observation -> validation -> incorporation. Negative examples include wrong-tool selection, malformed arguments, unsupported calls, unnecessary calls, and correct calls whose outputs are misinterpreted. ReAct [4], Toolformer [5], ReTool [11], and Agent Lightning [12] motivate different parts of this design, while MATI adds provenance, permissions, HyperSkill binding, and world-effect verification.

A tool should be learned as a contract rather than a magic token. The contract includes preconditions, side effects, reversibility class, expected observation schema, timeout/failure modes, and postconditions. During RL, reward is decomposed so that a successful final answer does not excuse an unauthorized or unverifiable call. Dynamic interleaving is important for mathematics and systems work: the model may reason, invoke code, inspect the result, revise a hypothesis, and only then continue the derivation.

Operationalization. For tool-use internalization, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for optimization because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target credit assignment: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out ablation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

13. HyperSkill Distillation

Page thesis. Defines a skill as a bounded executable policy object with trigger, preconditions, steps, evidence contract, outputs, failure modes, and promotion tests. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

A HyperSkill is represented as HS=(trigger, preconditions, inputs, policy, tools, evidence_contract, outputs, stop, failure_modes, tests, memory_delta). The compiler turns each skill into demonstrations, partial-completion exercises, repair tasks, trigger discrimination, and composition examples. Skill success is measured by postconditions and verifier tests, not by textual similarity to a reference answer.

Skill distillation supports abstraction. Multiple low-level skills can share a latent operator, such as retrieve-verify-cite, compare-rank-select, or plan-execute-checkpoint. The model is trained to recognize these abstractions while retaining exact skill boundaries. Promotion from a generated procedure to a canonical HyperSkill requires regression tests, adversarial cases, reversibility checks, and evidence that it reduces error or cost. This mirrors Knowledge-Integrity AutoDeployment: model-generated procedures are hypotheses until verified.

Operationalization. For hyperskill distillation, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for dataset engineering because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target provenance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out split isolation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

14. HyperLoop Distillation

Page thesis. Defines bounded loops as stateful control policies with stop conditions, watchdogs, verification gates, and memory writeback. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

A HyperLoop is a bounded recurrent policy HL=(state, body, verifier, watchdog, budget, stop_condition, rollback, writeback). Examples include search-until-evidence-sufficient, generate-test-repair, plan-act-observe, or multi-agent debate with a maximum number of rounds. Training traces expose both continuation and termination decisions. The most important label is often not the next step but whether another iteration is justified.

Loop failure modes include infinite continuation, oscillation, premature stopping, context bloat, unverified accumulation, and irreversible action before convergence. MATI introduces Loop Closure Reliability: the fraction of loops that stop within budget with verified postconditions satisfied. The model also learns checkpoint compression so that each iteration passes forward only the minimal state needed, supporting Context Genome and token-economy principles.

Operationalization. For hyperloop distillation, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for agent control because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target bounded execution: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out state verification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

15. HKOS and Memory Internalization

Page thesis. Separates parametric theory memory from external HyperKB retrieval and trains explicit retrieval, provenance, compaction, and mid-task injection behaviors. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

HKOS internalization is intentionally incomplete. Large, mutable, provenance-sensitive knowledge remains external. The model is instead trained to retrieve, interpret, compare, compress, and cite it. This separation follows the general motivation of RAG [7] but extends retrieval objects beyond passages to theory nodes, skills, tools, memories, and operational state. The model must learn when parametric recall is sufficient and when external truth should dominate.

Memory writes are delta-based. A candidate memory m is accepted only if it is task-relevant, non-duplicative, supported, appropriately scoped, and safe to persist. Self-Reconstruction Memory then rebuilds a compact state from verified deltas rather than appending entire transcripts. Training examples include memory rejection, contradiction reconciliation, stale-memory detection, and mid-task injection of newly retrieved facts. The objective is continuity without uncontrolled accumulation.

Operationalization. For hkos and memory internalization, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for evaluation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target calibration: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out falsification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

16. Multi-Agent and Cross-Model Distillation

Page thesis. Uses specialized teachers and routing traces to distill candidate comparison, disagreement resolution, and model/compute allocation into a student policy. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Cross-model distillation treats multiple models and agents as teachers with heterogeneous strengths. For each task, the dataset can store candidate outputs, route cost, latency, evidence quality, verifier scores, and final selection rationale. The student then learns both task behavior and meta-routing. Disagreement is not automatically resolved by majority vote; evidence-ranked arbitration can prefer a minority candidate if its proof or tool result is stronger.

A Multi-Agent Candidate Comparison loss trains pairwise and listwise selection across plans. A Teacher Reliability prior is updated by domain and task type. This creates a model-geo memory: the OS learns where a teacher is strong, weak, expensive, or unstable. The target student need not reproduce every teacher capability; it should learn when escalation is worth the compute and when local reasoning is adequate.

Operationalization. For multi-agent and cross-model distillation, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for representation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target semantic invariance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out graph perturbation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

17. Falsification Courts and Process Verification

Page thesis. Trains verifier-guided reasoning with contradiction, evidence-rank, no-loss, hallucination, reduction, and world-effect courts. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Falsification is implemented as trainable verification, not rhetorical criticism. Each reasoning step can be checked by one or more courts: contradiction, evidence rank, no-loss, hallucination, anti-reduction, synthetic efficiency, question efficiency, gap detection, world-effect, and constitutional safety. Process-verifier research [8] motivates progress-sensitive supervision; MATI adds domain-specific court labels and uses them for both search and training.

The Demolition Court generator creates adversarial variants by deleting assumptions, swapping causal direction, changing a sign, inserting unsupported evidence, violating a permission tier, or making a plausible but unverifiable claim. The target model must identify the failure and repair it. Court agreement can be calibrated against executable checks whenever possible. The verifier itself is subject to a meta-court to detect systematic false positives, reward hacking, or preference collapse.

Operationalization. For falsification courts and process verification, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for optimization because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target credit assignment: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out ablation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

18. RLVR, Process Rewards, and World-Effect Reward

Page thesis. Combines verifiable rewards with progress-sensitive process rewards and a verified state-change objective for agentic trajectories. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Outcome reward alone is insufficient for long agentic trajectories. MATI combines verifiable outcome rewards with step-level process progress. A proposed world-effect term is Delta_world(t) = V_verified(S_{t+1}) - V_verified(S_t), where V_verified only changes after an observation confirms the expected state transition. Text claiming success therefore receives no world-effect credit. For purely mathematical tasks, the analogous state change is verifier-confirmed advancement toward a valid solution.

RLVR is especially useful where exact correctness tests exist, but MATI decomposes reward into correctness, evidence support, routing efficiency, tool validity, authorization, and loop closure. The reward is clipped so that one high-scoring dimension cannot compensate for a severe constitutional violation. Credit assignment maps rewards backward to the question or routing decision that enabled progress. This structure is compatible with agent-training approaches that separate execution from learning [12].

Operationalization. For rlvr, process rewards, and world-effect reward, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for dataset engineering because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target provenance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out split isolation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

19. Constitutional Reversibility and Authorization Training

Page thesis. Makes permission tiers, reversibility, rollback, and irreversible-risk aversion first-class training targets. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Constitutional reversibility is encoded as action metadata. T0 actions are read-only reasoning; higher tiers correspond to increasing side effects and authorization requirements. Training data include near-identical pairs where the only difference is permission state, forcing the model to condition behavior on authorization rather than task desirability. Reversibility loss penalizes irreversible actions when a reversible probe or simulation could reduce uncertainty first.

Rollback is a first-class target. A successful agent must know the compensating action, backup requirement, or non-reversibility boundary before acting. This is particularly important when fine-tuning models for code execution, infrastructure changes, outreach, or file operations. The constitutional objective is not generic refusal: it is bounded agency with maximum useful action inside verified permissions.

Operationalization. For constitutional reversibility and authorization training, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for agent control because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target bounded execution: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out state verification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

20. End-to-End Training Pipeline

Page thesis. Specifies canonicalization -> SFT -> preference optimization -> process-verifier training -> RLVR -> tool/loop RL -> adversarial courts -> promotion. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

The end-to-end pipeline has twelve stages: (0) archive and register sources; (1) compile TheoryGraph; (2) generate semantic and mathematical transforms; (3) SFT for definitions, equations, and structured state; (4) SFT for questions, tools, skills, and loops; (5) DPO or equivalent preference optimization for question/routing/evidence choices; (6) train process verifiers; (7) RLVR on mathematical and executable tasks; (8) agent/tool RL on bounded environments; (9) multi-agent teacher distillation; (10) adversarial court training; (11) promotion only after held-out regression and safety tests.

Each stage writes a dataset manifest with source hashes, transformation version, theory-node IDs, and split membership. This makes experiments reproducible and permits rollback if a training batch introduces regressions. The preferred implementation uses modular adapters so a failure can be localized. The training OS maintains a lineage graph from each final adapter weight to the examples and canonical theory objects that influenced it.

Operationalization. For end-to-end training pipeline, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for evaluation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target calibration: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out falsification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

21. Parameter-Efficient Adaptation for Large and MoE Models

Page thesis. Explains LoRA/QLoRA adapter placement, rank allocation, MoE considerations, curriculum mixing, anti-forgetting replay, and adapter composition. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

For large models, LoRA [1] and QLoRA [2] provide the default parameter-efficient baseline. MATI recommends rank allocation by behavioral layer rather than uniform rank: attention projections may receive higher rank for routing and relation learning; selected MLP projections may receive rank for theory semantics; tool-format and governance behavior can be isolated in dedicated adapters. For Mixture-of-Experts models, experiments should separately measure shared attention adaptation and expert-specific drift.

A practical curriculum begins with a low learning rate and high-quality canonical SFT, then adds harder synthetic transformations and adversarial negatives. Replay from general instruction data reduces catastrophic forgetting. Adapter composition is evaluated before merging. Quantization-aware workflows should verify that symbolic fidelity, tool-call syntax, and calibration survive deployment quantization. Thinking-budget behavior, as described for Qwen3 [10], can be repurposed as an inference-time complement to learned Sparse HyperRouting.

Operationalization. For parameter-efficient adaptation for large and moe models, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for representation because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target semantic invariance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out graph perturbation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

22. AGILLM Evaluation Suite

Page thesis. Defines Theory Coverage Internalization, Governed Question Gain, Routing Regret, Evidence Calibration, Tool Utility, Loop Closure, Reversibility, and World-Effect metrics. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

The primary metric is TCI = weighted_correct_theory_relations / weighted_tested_theory_relations. TCI tests whether weighted theory relations remain correct under paraphrase, symbol renaming, compositional chaining, and adversarial distractors. Governed Question Gain measures whether generated questions increase expected downstream success. Routing Regret measures wasted or insufficient compute. Evidence Calibration compares confidence to source strength. Tool Utility measures net verified gain per call. Loop Closure Reliability measures bounded convergence. Constitutional Compliance measures authorization correctness without rewarding useless refusal.

World-Effect Precision is the proportion of claimed external state changes that are verified. Memory Reconstruction Fidelity measures whether persisted deltas are supported, non-duplicative, and sufficient for continuation. Cross-Model AGI Density can be operationalized as verified task utility per unit of active parameter compute, latency, or cost, although the exact normalization remains a research choice. The benchmark suite should report all dimensions rather than collapsing them into a single AGI score.

Operationalization. For agillm evaluation suite, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for optimization because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target credit assignment: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out ablation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

23. Experimental Design and Ablation Program

Page thesis. Proposes controlled experiments that isolate which MATI components cause generalization, reliability, and agentic competence. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

Ablation A compares document-only SFT against TheoryGraph-compiled SFT. Ablation B removes QMCD question ranking. Ablation C removes process-verifier rewards. Ablation D removes tool traces. Ablation E replaces external HKOS with parametric memorization. Ablation F removes constitutional pairs. Ablation G compares monolithic versus modular adapters. Ablation H varies synthetic-to-canonical data ratios. Ablation I evaluates routing learned from teacher traces versus rule-only routing. Ablation J tests whether Demolition Court adversarial data improves out-of-distribution theory transfer.

Key hypotheses are directional and falsifiable: graph compilation should improve compositional theory tests; question-governed training should reduce premature action on under-specified tasks; process rewards should improve sample efficiency on long trajectories; world-effect rewards should reduce false success claims; external HKOS should improve updateability and provenance at the cost of retrieval dependence; modular adapters should reduce destructive interference but may underperform merged joint training on tightly coupled tasks.

Operationalization. For experimental design and ablation program, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for dataset engineering because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target provenance: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out split isolation. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

24. Failure Modes, Limits, and Falsifiable Research Claims

Page thesis. Enumerates contamination, symbolic mimicry, reward hacking, policy collapse, tool overuse, retrieval dependence, and theory overfitting with explicit falsifiers. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

The largest risk is symbolic mimicry: the model may reproduce names such as MCD or HyperLoop without preserving their constraints. TheoryGraph perturbation tests are designed to expose this. A second risk is reward hacking: a model can learn superficial patterns that satisfy verifiers. Executable checks, adversarial meta-verifiers, and held-out transformations reduce but do not eliminate this problem. A third risk is theory overfitting, where internal research terminology displaces general reasoning ability; general-domain replay and external benchmarks are therefore mandatory.

Other risks include tool overuse, router collapse to a favorite model, context bloat, retrieval hallucination, stale memory, permission bypass, and overconfidence in internally proposed hypotheses. MATI is falsified as a useful framework if a simpler baseline achieves equal or better theory generalization, tool reliability, and governance with lower compute and data complexity. The research claim is not that every named component is necessary, but that the architecture makes their necessity empirically testable.

Operationalization. For failure modes, limits, and falsifiable research claims, the training compiler represents every example at three simultaneous levels: surface text, typed theory structure, and execution semantics. Surface text supplies linguistic diversity; typed structure records which definitions, equations, assumptions, and constraints must remain invariant; execution semantics record what the model should retrieve, calculate, ask, route, call, verify, or refuse. This separation is important for agent control because an output can be semantically plausible yet violate the relation that makes the source theory useful. Each generated example therefore carries a provenance pointer, transformation identifier, epistemic status, expected action type, and verifier family. The model is rewarded for the smallest sufficient correct behavior, not for reproducing the longest source passage.

Dataset pattern. A canonical item is expanded into a family of positive and negative instances. Positive variants include paraphrase, symbol renaming, compositional chaining, cross-domain transfer, retrieval-conditioned completion, tool-assisted checking, and recovery after an injected error. Negative variants target bounded execution: missing assumptions, swapped causal direction, stale evidence, wrong tool choice, gratuitous tool use, overconfident unsupported claims, unauthorized side effects, and false declarations of completion. Preference pairs rank candidates by relation fidelity, evidence quality, routing cost, reversibility, and verified progress. The same source item can consequently supervise SFT, preference optimization, process-verifier learning, or RLVR without losing its lineage.

Verifier and experiment contract. Every claimed competence on this page must survive held-out state verification. A training run is promoted only if it improves the target metric without unacceptable regression on general instruction following, mathematical reasoning, retrieval fidelity, and constitutional behavior. The baseline hierarchy begins with prompting and retrieval only, then document-level SFT, then TheoryGraph SFT, then question/routing supervision, then process and reward-based training. This ordering lets experiments estimate marginal value rather than attributing every gain to the full stack. Failed trials are retained as research evidence and converted into Demolition Court cases or regression tests for later runs.

Systems implication. MATI assumes that the most capable design will remain hybrid. Parameters should internalize stable abstractions and recurring control policies, while HKOS retains mutable knowledge, large evidence objects, detailed skill specifications, and provenance. Exact arithmetic or environment inspection should move to tools; long recurrent behavior should move to bounded HyperLoops; high-impact side effects should remain behind explicit authorization. Fine-tuning is successful when the model learns these boundaries and routes correctly across them. A model that merely recites the architecture but cannot choose the appropriate boundary has not internalized the operating system in the MATI sense.

25. Conclusion, Reproducibility Blueprint, and References

Page thesis. Summarizes MATI as a research program and lists the reproducibility artifacts and external literature used to ground the post-training design. MATI treats this component as a policy-learning problem with explicit provenance and falsification boundaries. The source architecture is never assumed correct merely because it is present in the corpus; canonical claims are tagged as axioms, hypotheses, engineering heuristics, or empirical observations. This separation is required so that fine-tuning does not convert speculative research statements into unconditional model beliefs.

MATI reframes fine-tuning as an operating-system internalization problem. The target is a hybrid cognitive policy capable of manipulating a formal theory graph, governing questions, allocating compute sparsely, using tools and external memory, composing skills and loops, respecting authorization, falsifying its own candidate paths, verifying world effects, and reconstructing only supported memory deltas. This is a more demanding objective than teaching the model the vocabulary of an agentic framework, but it also yields sharper tests.

Reproducibility blueprint. Release or preserve: source registry; TheoryGraph schema and snapshots; transformation code; training manifests; split hashes; SFT/preference/RL datasets; verifier definitions; adapter configs; checkpoints; evaluation harness; court prompts/tests; tool sandbox; memory benchmark; and experiment ledger. Every claimed gain should be accompanied by an ablation and a held-out falsification test.

References

- [1] Hu et al. (2021), LoRA: Low-Rank Adaptation of Large Language Models, arXiv:2106.09685.
- [2] Dettmers et al. (2023), QLoRA: Efficient Finetuning of Quantized LLMs, arXiv:2305.14314.
- [3] Rafailov et al. (2023), Direct Preference Optimization: Your Language Model is Secretly a Reward Model, arXiv:2305.18290.
- [4] Yao et al. (2022), ReAct: Synergizing Reasoning and Acting in Language Models, arXiv:2210.03629.
- [5] Schick et al. (2023), Toolformer: Language Models Can Teach Themselves to Use Tools, arXiv:2302.04761.
- [6] Shinn et al. (2023), Reflexion: Language Agents with Verbal Reinforcement Learning, arXiv:2303.11366.
- [7] Lewis et al. (2020), Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, arXiv:2005.11401.
- [8] Setlur et al. (2024), Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning, arXiv:2410.08146.
- [9] DeepSeek-AI (2025), DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, arXiv:2501.12948.
- [10] Yang et al. (2025), Qwen3 Technical Report, arXiv:2505.09388.
- [11] Feng et al. (2025), ReTool: Reinforcement Learning for Strategic Tool Use in LLMs, arXiv:2504.11536.
- [12] Luo et al. (2025), Agent Lightning: Train ANY AI Agents with Reinforcement Learning, arXiv:2508.03680.
- [13] Wen et al. (2025), Reinforcement Learning with Verifiable Rewards Implicitly Incentivizes Correct Reasoning in Base LLMs, arXiv:2506.14245.

Internal theory lineage

- [M1] MOUSSAIF internal corpus: Vascular Questioning Core Ontology - MCD, Q-MCD, MCAI and HAKIM OS.
- [M2] MOUSSAIF internal corpus: FATTAH Agentic AGI HyperKB.
- [M3] MOUSSAIF internal corpus: MICAI / NeoAGI / QMCD research series.
- [M4] MOUSSAIF internal corpus: HyperSkills, HyperKB, HyperLoops state-of-the-art architecture.
- [M5] MOUSSAIF internal corpus: Agentic Governance Closure and constitutional reversibility stack.