MCARG: إطار حوكمة الهوية المرجعية ومنع الازدواج في أنظمة الذكاء الاصطناعي الوكيلية طويلة الأمد

2026

المحتويات

الملخص 4
1. مقدمة 5
2. المشكلة البحثية 5
3. الفرضية والمنهج 6
4. نموذج الهوية المرجعية 6
5. Invariant Zero-Duplication 6
6. Protocol ID Canonical 7
7. Engine Resolution Duplicate 7
8. Protocol Artfiact Canonical 7
9. Provenance وLineage 8
10. Versioning وMerge Semantic 8
11. Commit Transactional وConcurrency 8
12. Plane Registry 9
13. Sefl-Reconstruction 9
14. التكامل مع الذاكرة وGraphRAG 9
15. Governance Sefl-Improvement AGI-oriented 9
16. التدريب 10
17. التقييم وMCARG-Bench 10
18. الأمن والحوكمة 10
19. حدود الإطار 11
20. الخاتمة 11
مراجع مختارة 11

الملاحق الهندسية الموسعة **12**
ملحق: SPEC MASTER 01 12
1. تعريف النظام 12
2. الثوابت العليا 12
3. عقد الكتابة 12
4. مجالات الحوكمة 12
5. نواتج إلزامية 13
ملحق: MODEL FORMAL 03 13
1. الكون المحكوم 13

شرط التفرد 13
تعدد التمثيل 13
2. دالة التشابه المركبة 13
3. قرار الإجراء 14
4. فضاء القرارات 14
5. معيار canonicality 14
ملحق: ONTOLOGY 04 15
الأنواع الأساسية 15
العلاقات المعيارية 15
الفصل الابستيمي 15
البعد الزمني 15
ملحق: SCHEMAS 05 16
Schema Object Canonical 16
Record Registry 16
record Event 17
ملحق: PROTOCOL ID CANONICAL 06 17
الهدف 17
الصيغة 17
قواعد 17
Resolution Identity 18
حالات عدم اليقين 18
ملحق: INVARIANT DUPLICATION ZERO 07 18
Invariant Zero-Duplication — ZDI 18
ما لا يعنيه ZDI 18
فئات الازدواج 19
قاعدة الأمان 19
ملحق: ENGINE RESOLUTION DUPLICATE 08 19
Engine Resolution Duplicate — DRE 19
مرحلة generation candidate 19
التصنيف 19
thresholds افتراضية تجريبية 20
قاعدة contradiction 20
ملحق: PROTOCOL ARTIFACT CANONICAL 09 20
Protocol Artfiact Canonical — CAP 20
البوابات السبع 20
pseudo-flow 20
مخرجات CAP 21
ملحق: GRAPH LINEAGE PROVENANCE 10 21
Graph Lineage & Provenance — PLG 21
عقدة lineage 21
خواص واجبة 21
العلاقات 22
reconstruction 22
ملحق: PROTOCOL MERGE VERSION 11 22
Protocol Merge & Version — VMP 22
rule version 22
merge three-way semantic 22
قواعد 23

concurrency optimistic 23
ملحق: 12 PROTOCOL COMMIT REGISTRY 23
Protocol Commit Registry — RCP 23
transaction 23
objective atomicity 23
idempotency 24
ملحق: 13 RECONSTRUCTION SELF 24
Sefl-Reconstruction Integrity — ISR 24
الهدف 24
sequence boot 24
مصادر الحقيقة 24
invariant recovery 25
ملحق: 14 PACK AGENT 25
Pack Agent MCARG 25
duties of separation 25
contract message 25
ملحق: 15 PACK HYPERSKILL 26
S001-S060 HyperSkills 26
contract skill 27
ملحق: 16 PACK HYPERLOOP 28
HyperLoops 28
Deduplication Continuous L01 28
Integrity Registry L02 28
Sefl-Reconstruction L03 28
Consolidation Knowledge L04 28
Verfiication Provenance L05 28
Evolution Capability L06 28
Maintenance Truth Temporal L07 28
Recovery Orphan L08 28
termination 28
ملحق: 17 BENCHMARK EVALUATION 28
MCARG-Bench 28
مجموعات المهام 28
metrics 29
design benchmark 29
ملحق: 18 PACK TRAINING 30
تدريب الهوية والحوكمة 30
trajectories SFT 30
training contrastive 30
perturbation synthetic 30
objective preference 30
curriculum tool-use 30
ملحق: 19 RECOVERY FAILURE 31
Taxonomy Failure 31
pipeline repair 31
quarantine-first 31
ملحق: 20 GOVERNANCE SECURITY 32
Model Security 32
controls 32

gate world-effect 32
actions destructive 32
ملحق: 21 API RUNTIME 33
API Runtime 33
CLI 33
envelope response 34
ملحق: 22 SUITE TEST 34
Strategy Test 34
Unit 34
Property-based 34
Integration 34
Chaos 35
targets Acceptance 35
ملحق: 23 IMPLEMENTATION REFERENCE 35
Architecture Reference Minimal 35
contract storage technology-neutral 35
pseudo-code 36
ملحق: 24 ARCHITECTURE DEPLOYMENT 36
Profiles Deployment 36
single-user / Local 36
lab research / Team 36
runtime AGI Distributed 36
consistency 37
policy availability 37
ملحق: 25 OBSERVABILITY METRICS 37
Metrics 37
Health Global 37
events observability 37
dashboards 38
ملحق: 26 PLAYBOOK MIGRATION 38
MCARG to Migration 38
cleanup destructive Freeze — 0 Phase 38
Inventory — 1 Phase 38
Fingerprinting — 2 Phase 38
clusters Candidate — 3 Phase 38
assignment Canonical — 4 Phase 38
bacfkill Provenance — 5 Phase 38
build Registry — 6 Phase 38
build Graph — 7 Phase 38
Validation — 8 Phase 39
Cutover — 9 Phase 39
duplicates legacy Archive — 10 Phase 39

الملخص

تقدم هذه الورقة MCARG بوصفه إطاراً هندسياً ونظرياً لحوكمة الحالة الدائمة في الأنظمة الوكيلية طويلة الأمد والأنظمة الموجهة نحو الذكاء العام الاصطناعي. تنطلق الفرضية من أن توسع الأنظمة التي تنشئ المعرفة والمهارات والمهام والملفات تلقائياً يولد فئة من أعطال الاستمرارية لا تعالجها دقة الاستدلال وحدها: تعدد الهويات المرجعية لنفس الكيان، تضارب النسخ، فقدان المصدر

والنسب، الكتابة المتزامنة، والاعتماد على ذاكرة جلسة غير موثقة. يقترح MCARG فصلاً صريحاً بين التمثيل والهوية، ويعرف مبدأ Invariant Zero-Duplication بحيث يسمح بتعدد الصيغ والإصدارات والترجمات مع منع تعدد السلطات المرجعية غير المنضبطة داخل النطاق نفسه. يدمج الإطار حل الهوية المرجعية، محركاً دلالياً لتصنيف العلاقات، بروتوكولاً للartfiacts، رسم provenance/lineage، تحكماً في الإصدارات والدمج، commit ذرياً، وطبقة لإعادة بناء الحالة بعد فقد السياق أو تلف الفهارس. كما تقدم الورقة نموذجاً رسمياً، خوارزمية قرار، تصنيفاً للأعطال، معايير تقييم، وبرنامجاً تدريبياً لنماذج اللغة على مهارة التمييز بين duplicate وversion وtranslation وfork وcontradiction. الهدف النهائي هو تحويل الذاكرة الدائمة من تراكم ملفات إلى نظام هوية قابل للتدقيق وإعادة البناء.

الكلمات المفتاحية: Agentic MCARG، AI، Canonical AGI، Identity، Zero-Duplication، Provenance، Semantic Lineage، Versioning Knowledge، Registry Graph، Governance، Sefl-Reconstruction Long-Horizon. Agents

1. مقدمة

تتحول أنظمة الذكاء الاصطناعي المعاصرة تدريجياً من نمط سؤال/جواب قصير العمر إلى أنظمة تنفذ أهدافاً طويلة الأفق، تستخدم الأدوات، تعدل الملفات، تبني الذاكرة، وتنتج قرارات ونسخاً جديدة من المعرفة. في هذا السياق لا يكفي تقييم جودة الاستجابة النصية، لأن النظام قد ينجح محلياً في مهمة واحدة ثم يضعف عالمياً بسبب تراكم حالات متنافسة. نموذج يستطيع إنشاء تقرير جيد لكنه ينشئ عشر نسخ متعارضة من التقرير نفسه، أو يغير نظرية دون حفظ مصدرها، لا يملك استمرارية معرفية موثوقة.

مشكلة الازدواج هنا أعمق من تشابه bytes أو filenames. قد توجد نسخة Markdown وPDF وترجمة عربية وملخص تنفيذي وعرض تقديمي لنفس الكيان؛ هذه ليست أخطاء ما دامت مرتبطة بهوية واحدة. وفي المقابل قد توجد وثيقتان مختلفتان لفظياً لكنهما تعلنان نفسيهما مرجعاً نهائياً لنفس النظرية، وهنا يظهر تضارب في السلطة المرجعية حتى لو لم يكن النص متطابقاً.

تقترح هذه الورقة MCARG باعتباره kernel governance يفصل بين قدرة النموذج على اقتراح التغيير وبين سلطة إدخاله في الحالة الدائمة. وبدلاً من المسار البسيط `goal → generate → save` يصبح المسار `goal → identity resolve → provenance preserve → relationship classify → delta compute → state canonical retrieve → audit → commit`. هذا التغيير يحول التخزين من وظيفة لاحقة إلى جزء من الاستدلال النظامي.

2. المشكلة البحثية

يمكن تلخيص المشكلة في أربعة أنواع من الانجراف. الأول **Drfit Identity**، عندما يتغير الاسم أو المكان أو الصيغة فيعامل النظام الكيان ذاته ككيان جديد. الثاني **Drfit Authority**، عندما تصبح عدة نسخ مرجعاً فعالاً دون قاعدة تحدد الأحدث أو الأوثق. الثالث **Drfit Provenance**، عندما تبقى النتيجة وتختفي سلسلة الأدلة والتحولات التي أنتجتها. والرابع **Drfit State**، عندما تختلف السجلات والفهرس والرسم البياني ومخزن الملفات حول ما هو قائم فعلياً.

هذه الأعطال تتفاقم مع تعدد agents. يستطيع وكيلان قراءة النسخة 17 في الوقت نفسه، ثم يحاول كل منهما الكتابة. إذا لم توجد آلية expected-version أو merge، يمكن أن تمحو الكتابة الثانية الأولى. كذلك قد يعيد agent تنفيذ العملية نفسها بعد timeout، فينشئ duplicate أو يكرر side effect. ولذلك تتقاطع الحوكمة مع مفاهيم control concurrency وidempotency وtransaction وlogging.

المسألة الثالثة هي أن التشابه لا يساوي الهوية. قد يكون نصان متشابهين لأنهما يناقشان موضوعاً واحداً لكنهما متناقضان في الاستنتاج. حذف أحدهما باعتباره duplicate يدمر المعلومة الابستيمية. وقد يكون نصان مختلفين لغوياً لأن أحدهما ترجمة للآخر؛ تجاهل العلاقة ينتج ازدواجاً دلالياً. يحتاج النظام إذاً إلى **classfiication relationship** لا binary deduplication فقط.

3. الفرضية والمنهج

الفرضية المركزية هي أن استمرارية النظام الوكيلي تتحسن إذا فصلنا بين `identity`, `representation`, `authority`, `version`, و`provenance` وجعلنا كل effect world يمر عبر بوابة تحقق. وتبني الورقة منهج design-science: تعريف invariants، بناء model نظري، تحويله إلى protocols، ثم اقتراح benchmark واختبارات falsfiication يمكن أن تكشف فشل النموذج.

لا تزعم الورقة أن MCARG يحقق AGI أو يقدم برهاناً على وعي أو ذكاء عام. هو طبقة حوكمة هندسية مصممة لخاصية أضيق لكنها مركزية: الحفاظ على اتساق الهوية والحالة عبر زمن طويل، عدة agents، عدة representations، وتغييرات متكررة. لذلك ينبغي تقييمه بمقاييس integrity وreconstruction وpreservation information لا بمقاييس القدرات العامة فقط.

4. نموذج الهوية المرجعية

ليكن E مجموعة الكيانات المحكومة، وR مجموعة التمثيلات، وS مجموعة النطاقات، وT الزمن. نعرف دالة `C(e,s,t)` التي تعيد الحالة المرجعية الفعالة لكيان e داخل النطاق s والزمن t. يفرض MCARG أن عدد الحالات المرجعية الفعالة لا يتجاوز واحدة ضمن scope واحد، مع السماح لفروع scopes مختلفة مثل research وproduction وexperimental.

$$\forall e, s, t : |C(e, s, t)| \leq 1$$

أما التمثيلات فتتبع:

$$Representations(e) = \{r_1, r_2, ..., r_n\},\ n \geq 1$$

هذا الفصل يسمح بوجود PDF وMarkdown وJSON وembedding وترجمة وملخص دون اعتبارها هويات مستقلة. كل تمثيل يملك relation مثل `EXPORT_OF`, `TRANSLATION_OF`, أو `VIEW_OF`.

5. Invariant Zero-Duplication

يعرف ZDI كالتالي: إذا كان x وy حالتين مرجعيتين فعالتين داخل النطاق والزمن نفسيهما، وكانت هويتهما الدلالية واحدة، فيجب أن يكونا الحالة المرجعية نفسها.

$$Identity(x) = Identity(y) \wedge x, y \in ActiveCanonical(s, t) \Rightarrow x = y$$

الهدف ليس إزالة كل duplicate bytes. النسخ الاحتياطية والتصديرات والفهارس المشتقة قد تتكرر مادياً. المقصود منع

identities canonical duplicate uncontrolled. ومن ثم فإن ZDI خاصية حوكمة للسلطة المرجعية، وليس سياسة اقتصاد مساحة تخزين.

6. Protocol ID Canonical

لكل كيان معرف immutable من الشكل:

`MCARG:<TYPE>:<NAMESPACE>:<SEQUENCE>`.

يجب ألا يتغير المعرف عند rename أو move أو export. إذا انقسم كيان إلى كيانين، تنشأ هويات جديدة مرتبطة بـ`DERIVED_FROM`. وإذا اندمج كيانان، لا تحذف IDs القديمة؛ بل توجه بعلاقة `MERGED_INTO` إلى المرجع الناتج. بهذه الطريقة تبقى الروابط التاريخية قابلة للتفسير حتى بعد إعادة التنظيم.

حل الهوية يبدأ IDs exact ثم aliases، ثم lexical/semantic/graph/provenance candidates. إذا كان المرشح الأفضل غير حاسم يجب إرجاع `UNRESOLVED_IDENTITY` أو quarantine بدلاً من فرض اختيار. هذا القرار بالغ الأهمية لأن خطأ resolution identity ينتشر في كل السجلات اللاحقة.

7. Engine Resolution Duplicate

يقترح DRE تمثيل التشابه كدالة مركبة:

$$D(A,B) = w_s S + w_l L + w_t T + w_g G + w_p P + w_r R + w_h H$$

حيث تمثل الحدود semantic، lexical، structural، graph، provenance، role وhash similarities، مع مجموع الأوزان واحداً. النتيجة ليست حكماً نهائياً بالهوية؛ بل vector evidence يستخدمه classfiier وpolicy. engine

يستخدم التصنيف فئات D0 إلى D8: binary duplicate، formatting duplicate، semantic duplicate، version، partial overlap، derived translation، artfiact، fork وfalse duplicate. ويمكن إضافة `CONTRADICTION` كفئة معرفية متعامدة عندما يكون الموضوع واحداً والمضمون متعارضاً.

القيمة الحقيقية للمحرك في تقليل نوعين من الأخطاء: merge false الذي يدمر اختلافاً حقيقياً، وsplit false الذي ينشئ هويتين لنفس الكيان. لذلك يجب معايرة thresholds على corpus فعلي وقياس precision/recall لكل relation على حدة.

8. Protocol Artfiact Canonical

CAP هو مسار سبع بوابات: Existing-State Identity، Dffierence Search، Relationship Extraction، Provenance Classfiication، Canonical Attachment، Integrity Commit. Verfiication. بهذه البنية تصبح الكتابة عملية محكومة وليست effect side مباشر للنموذج.

إذا كان proposal مطابقاً لحالة قائمة، يعيد reference CAP بدلاً من إعادة الإنشاء. إذا كان alias يسجله في ALIAS_REGISTRY. إذا كان version يحسب delta ويسجل parent. إذا كان contradiction يحفظ الطرفين ويضيف edge. وإذا كانت هوية جديدة حقيقية فقط عندها ينشئ object canonical جديداً.

هذه القاعدة تعيد تعريف الكفاءة. النظام الأفضل ليس الذي ينتج أكبر عدد من الملفات، بل الذي يحقق أكبر **gain information** بأقل fragmentation. يمكن التعبير عن ذلك بصورة تقريبية عبر الفرق بين المعرفة بعد وقبل الفعل، مع عقوبة للـduplication/conflict.

9. Provenance وLineage

الـprovenance يصف الأصل والأدلة؛ أما lineage فيصف التحول عبر الزمن. MCARG يحتاج كليهما لأن معرفة المصدر دون تاريخ التعديلات لا تكفي، ومعرفة الإصدارات دون مصادر الادعاءات لا تكفي أيضاً.

كل mutation مهم يسجل actor، operation، parents، sources، reason، timestamp content. transaction hash، ID. ويجب أن يكون سجل lineage منطقياً append-only، بحيث يصحح الخطأ بإضافة event correction/supersede لا بحذف التاريخ.

عند دمج وثيقتين يجب حفظ provenance على مستوى الادعاء متى كان ذلك عملياً، لا على مستوى الوثيقة فقط. بهذه الطريقة لا يؤدي merge شامل إلى جعل جميع الأدلة تبدو وكأنها تدعم جميع الجمل.

10. Versioning وMerge Semantic

التحكم في النسخ ليس ترقيم الملفات فقط. MCARG يفرق بين تغيير rendering وبين تغيير دلالي. يمكن لتصدير PDF جديد أن يكون representation جديداً دون رفع النسخة الدلالية، بينما تعديل فرضية أو معادلة يتطلب event. version

عند وجود فرعين يستخدم merge three-way semantic مع الجد المشترك C. تصنف الوحدات إلى unchanged، added، removed، conflict، contradiction. superseded. الدمج لا يكون concatenation آلياً، لأن الجمع الخام قد يكرر المحتوى أو يخفي تعارضاً.

يجب أن يحقق merge هدف حفظ المعلومة:

$$I(Merge(A,B)) \geq I(Unique(A \cup B)) - \epsilon$$

حيث صغير ومفسر، ويقتصر على التكرار أو الشكل غير الدلالي. إذا لم يمكن التحقق من ذلك، يفضّل النظام quarantine أو review. human

11. Transactional Commit وConcurrency

في نظام متعدد agents يجب فصل proposal عن commit. يبدأ RCP بقراءة expected version، ثم dffi وpolicy validation، ثم يكتب artfiact وprovenance والregistry والgraph ضمن transaction واحدة أو محاكاة transaction باستخدام write-ahead log وcommit marker.

الشرط الرئيسي هو منع الحالة الجزئية. إذا كتب الملف ولم يحدث registry، فقد يصبح artfiact يتيماً. وإذا حدث registry قبل نجاح التخزين يصبح pointer مكسوراً. لذلك يجب أن يكون idempotent replay وأن يحمل كل effect world `idempotency_key`.

التحكم المتفائل في التزامن بسيط: A Agent وB يقرآن version 17؛ إذا التزم A بـ18، يجب أن يفشل commit الخاص بـB بسبب expected-version mismatch ثم يعيد rebase/merge. overwrite الصامت محظور.

12. Plane Registry

يحول MCARG المعرفة إلى plane control عبر سجلات: ENTITY، ARTIFACT، SOURCE، VERSION، EXPERIMENT، DECISION، RELATIONSHIP_GRAPH وGLOBAL_CHANGELOG، كما يوجد registries. CAPABILITY، AGENT، SKILL، PROJECT، ALIAS، PROVENANCE، LATEST_STATE،

ليست كل السجلات مصادر حقيقة بالدرجة نفسها. يمكن اعتماد ترتيب: log event غير القابل للتلاعب، registries، lineage، content canonical graph، store ثم indexes/caches. هذه الأولوية مهمة عند التعارض؛ index vector مثلاً مشتق ويمكن إعادة بنائه ولا ينبغي أن يقرر canonical truth.

13. Sefl-Reconstruction

أحد أهداف MCARG أن يبقى النظام قابلاً للاستمرار بعد فقد جلسة النموذج أو تبديله. يبدأ boot بتحميل master، registries فحص schema/hash، إعادة latest pointers، بناء graph، بناء search/vector indexes، كشف orphan/duplicate، ثم استعادة goals/agents/skills والقيام بتدقيق شامل.

يمكن تعريف Fidelity Reconstruction كتشابه الحالة المستعادة مع آخر حالة صالحة:

$$RF = Similarity(State_{reconstructed}, State_{last_valid})$$

الهدف المثالي RF قريب من واحد، مع تقرير واضح لكل عنصر غير قابل للاسترجاع. بذلك تصبح الاستمرارية خاصية في النظام الخارجي لا في ذاكرة النموذج المؤقتة.

14. التكامل مع الذاكرة وGraphRAG

في RAG تقليدي قد تسترجع قاعدة المتجهات chunks من نسخ قديمة وحديثة معاً، فتخلق سياقاً متناقضاً. MCARG يقترح أن يبدأ retrieval من resolution entity ثم neighborhood graph canonical ثم active، state وبعدها يستدعي الإصدارات التاريخية أو الأدلة إذا احتاج السؤال.

المسار يصبح: `query → entity resolve → state canonical → neighborhood graph → evidence/history → retrieval semantic → assembly context`. بهذه البنية لا يختفي التاريخ، لكنه لا ينافس الحالة المرجعية الحالية بلا تمييز.

15. Governance Sefl-Improvement AGI-oriented

إذا سمحنا للنظام بإنشاء skill محسنة، لا ينبغي أن تصبح مهارة جديدة غير مرتبطة تلقائياً. يجب حل capability الحالية، حساب delta، بناء candidate، version، تقييمه مقابل benchmark، ثم promotion أو rejection مع lineage. يمكن قياس فرق الأداء:

$$\Delta P = P(v_{n+1}) - P(v_n)$$

ولا يرقّى الإصدار إذا لم يحقق threshold أو إذا أدخل regression يتجاوز حدود policy. هكذا يصبح recursive improvement عملية قابلة للتدقيق بدلاً من انفجار غير منضبط للمهارات.

16. التدريب

يمكن تدريب النماذج على MCARG عبر trajectories: supervised، duplicate□alias، revision□version، idea□create. new translation□link، both، contradiction□preserve ثم pairs contrastive صعبة تفرق بين نفس المفهوم ومفهوم قريب، وبين fork وupdate، وبين summary وreplacement.

كما يمكن بناء corpus synthetic بأخذ artfiact مرجعي وإجراء، rename، translation، paraphrase، reorder، split. العلاقة الأصلية معروفة، ما يوفر بيانات تدريب واسعة لاختبار reformat. fork، contradiction، expansion، merge، reasoning. identity

المكافأة في optimization preference يجب أن تفضل correct duplication، minimal provenance، reuse، preservation، information linking، state، persistent fabricated lineage، lost overwrite، silent وتعاقب merge. destructive

17. التقييم وMCARG-Bench

تتضمن المقاييس Duplicate Rate، Orphan Coverage، Provenance Coverage، Canonical Rate، Preservation. Information Fidelity، Reconstruction ويجب إضافة precision/recall لكل relation لأنه قد يحقق النظام accuracy عالية بإفراطه في فئة DISTINCT.

اختبارات benchmark تشمل exact/semantic duplicate، version، translation، fork، contradiction، merge، temporal writes، concurrent reconstruction، context-loss recovery، orphan retention، provenance identity. cross-format وcollision، alias adversarial truth،

من المهم بناء سيناريوهات falsfiication، مثل حالتين شديدتي التشابه لكن استنتاجهما متضاد، أو ملفين بنفس الاسم لكن لهما lineage مختلفة. نجاح النظام في الحالات السهلة لا يثبت سلامته في cases edge التي تسبب corruption فعلياً.

18. الأمن والحوكمة

تظهر هجمات خاصة بطبقة الهوية: alias، poisoning hijacking عبر ملفات أسماءها مقاربة، replay لعمليات قديمة، rollback إلى stale version، fabricated provenance، أو malicious. merge لذلك يجب أن تتكامل MCARG مع RBAC/ABAC، signed/hashed commit، append-only events، audit، idempotency، expected-version، وpolicy-as-code.

قاعدة gate world-effect تعني أن agent worker لا يكتب مباشرة إلى canonical. store يمر المسار عبر resolver ثم guardian policy ثم controller writeback ثم auditor. في تغييرات عالية الخطورة يفضل duties of separation أو approval. human

19. حدود الإطار

MCARG لا يحل صدق المعرفة من تلقاء نفسه. canonicality قد تجعل مصدراً ما مرجعاً تشغيلياً لكنها لا تثبت أنه صحيح علمياً. ولذلك يجب فصل authority عن truth، وإبقاء trust verfiication/confidence/source كخصائص مستقلة.

كذلك فإن similarity semantic وclassfiiers LLM-based قابلة للخطأ والانحياز. يجب ألا تتحول درجات التشابه إلى قرارات حذف غير قابلة للعكس. التصميم السليم يستخدم evidence، thresholds aggregation قابلة للمعايرة، quarantine، audit، وrollback.

وأخيراً، تكلفة الحوكمة ليست صفراً. كل write يضيف lookup وdffi وcommit وlogging. يجب موازنة consistency والlatency بحسب risk، class؛ قد تكون writes المؤقتة منخفضة المخاطر أخف، بينما research/theory canonical state يحتاج مساراً كاملاً.

20. الخاتمة

يقترح MCARG إعادة تصور منع الازدواج كحوكمة للهوية والسلطة المرجعية، لا كتنظيف للمحتوى. القيمة الرئيسية هي فرض علاقة واضحة بين الكيان وتمثيلاته، وحماية التاريخ والمصدر، وفصل الاقتراح عن commit، وتمكين النظام من إعادة بناء نفسه من سجلات خارج ذاكرة النموذج.

يمكن تلخيص العقد التنفيذي في العبارة: **before dffi name; before resolve create; before Search before verfiy merge; before preserve duplicate; before link copy; before version regenerate; effect. world every after register commit;** إذا أثبتت التجارب أن هذه المبادئ تخفض الازدواج غير المنضبط، تزيد fidelity عند reconstruction، وتقلل loss information في الدمج، فسيمثل MCARG طبقة عملية مهمة للأنظمة الوكيلية طويلة الأمد والأنظمة الموجهة نحو AGI.

مراجع مختارة

1. .*American Scientfiic* Web. Semantic The .(2001) O. Lassila, & J., Hendler, T., Berners-Lee,
2. *and Compression* documents. of containment and resemblance the On .(1997) Z. A. Broder, .*Sequences of Complexity*
3. crawling. web for near-duplicates Detecting .(2007) D. A. Sarma, & A., Jain, S., G. Manku, .*WWW*
4. Recommendation. W3C Model. Data PROV The PROV-DM: .(2013) al. et L., Moreau,
5. Morgan .*Techniques and Concepts Processing*: *Transaction* .(1992) A. Reuter, & J., Gray, Kaufmann.
6. .*Surveys Computing ACM* Graphs. Knowledge .(2021) al. et A., Hogan,
7. NLP Knowledge-Intensive for Generation Retrieval-Augmented .(2020) al. et P., Lewis, .*NeurIPS* Tasks.
8. .*ICLR* Models. Language in Acting and Reasoning Synergizing ReAct: .(2023) al. et S., Yao,
9. Tools. Use to Themselves Teach Can Models Language Tooflormer: .(2023) al. et T., Schick, .*NeurIPS*
10. Learning. Reinforcement Verbal with Agents Language Reflexion: .(2023) al. et N., Shinn,

.*NeurIPS*

.11 Park, J. S., et al. (2023). Generative Interactive Agents of Simulacra of Human Behavior. *UIST*.

الملاحق الهندسية الموسعة

تضم الملاحق التالية المواصفة التنفيذية المكملة للورقة، بحيث يمكن الانتقال من النموذج النظري إلى contract implementation واختبارات تشغيلية.

ملحق: SPEC MASTER 01

1. تعريف النظام

MCARG هو **Kernel Governance** يفصل بين قدرة النموذج على الاقتراح وبين سلطة تغيير الحقيقة الدائمة للنظام. كل كتابة دائمة تمر عبر حل الهوية، فحص الحالة الحالية، استخراج الفرق، تصنيف العلاقة، حفظ المصدر والنسب، ثم commit ذري وتدقيق لاحق.

2. الثوابت العليا

- **Invariant: Identity Canonical** لكل كيان محكوم معرف غير قابل لإعادة الاستخدام.
- **Invariant: Zero-Duplication** لا يوجد مرجعان فعّالان متنافسان للهوية نفسها ضمن النطاق ذاته.
- **Scope: Per Authority Single** مرجع فعّال واحد لكل `(entity, scope, time)`.
- **Provenance: Immutable** كل mutation مهم يترك أثراً يشرح المصدر، الأب، الفاعل، السبب، والتوقيت.
- **Destruction: Silent No** الاستبدال الصامت محظور؛ استخدم supersede/archive/retract.
- **Copy: Before Alias** الاسم الثاني alias، لا نسخة مرجعية جديدة.
- **Reinvention: Before Version** التطور version، لا `final-final-v2`.
- **Creation: Before Dffierence** أنشئ الفرق المفيد، لا المحتوى الموجود أصلاً.
- **Duplication: Not Is Contradiction** التناقض علاقة معرفية يجب حفظها.
- **Reconstructibility:** يجب أن تكون الحالة الدائمة قابلة لإعادة البناء من السجلات والـlineage.

3. عقد الكتابة

```
PROPOSAL != COMMIT
MEMORY CLAIM != VERIFIED PERSISTENT STATE
MODEL CONFIDENCE != AUTHORITY
SIMILARITY != IDENTITY
```

4. مجالات الحوكمة

MCARG يحكم: النظريات، الادعاءات، الأبحاث، artfiacts، المشاريع، prompts، agents، skills، tools، workflows، datasets، experiments، source records، decisions، policies، code components، configurations، deployments، checkpoints، والنتائج التشغيلية.

5. نواتج إلزامية

كل commit يجب أن ينتج أو يحدّث، بحسب الحاجة:

```
ENTITY_REGISTRY
ARTIFACT_REGISTRY
SOURCE_REGISTRY
VERSION_REGISTRY
LATEST_STATE_REGISTRY
PROVENANCE_REGISTRY
RELATIONSHIP_GRAPH
GLOBAL_CHANGELOG
```

وعند وجود خلل:

```
DUPLICATE_REPORT
ORPHAN_REPORT
CONFLICT_REPORT
DATA_INTEGRITY_REPORT
UNRESOLVED_ITEMS
```

ملحق: MODEL FORMAL 03

1. الكون المحكوم

ليكن E مجموعة الكيانات، وR مجموعة التمثيلات، وS مجموعة النطاقات، وT الزمن، و`C(e,s,t)` دالة تعيد الهوية المرجعية الفعّالة لكيان e في النطاق s والزمن t.

شرط التفرد

$$\forall e, s, t: \quad |C(e, s, t)| \leq 1$$

تعدد التمثيل

$$Representations(e) = \{r_1, ..., r_n\},\ n \geq 1$$

ولا يخرق ذلك ZDI ما دامت كل r_i مرتبطة بـcanonical ID واحدة.

2. دالة التشابه المركبة

لزوج مرشحين A,B:

$$D(A,B) = w_s S + w_l L + w_t T + w_g G + w_p P + w_r R + w_h H$$

حيث:

- S: similarity، semantic
- L: similarity، lexical
- T: similarity، structural
- G: similarity، graph-neighborhood
- P: similarity، provenance
- R: similarity، role/purpose
- H: similarity، content/hash

مع:

$$\sum_i w_i = 1$$

التشابه لا يثبت الهوية وحده، بل يولد **مرشح resolution**.

3. قرار الإجراء

$$a^* = \arg\max_a [N(a) + I(a) + U(a) - D_c(a) - L(a) - K(a)]$$

حيث N مكسب الجدة، I مكسب السلامة، U المنفعة، D_c تكلفة الازدواج، L فقد المعلومة، وK خطر التعارض، مع القيود:

$$IdentityPreserved = True$$

$$ProvenancePreserved = True$$

$$PolicySatisfied = True$$

4. فضاء القرارات

```
CREATE | UPDATE | VERSION | MERGE | ALIAS | LINK | FORK |
CONTRADICTION | REJECT | QUARANTINE
```

5. معيار canonicality

$$C = \alpha A + \beta V + \gamma P + \delta R + \epsilon F$$

الأوزان completeness. :F relevance، temporal :R quality، provenance :P verfiication، :V authority، :A policy-configurable ولا تعد ثوابت علمية.

ملحق: ONTOLOGY 04

الأنواع الأساسية

```
Knowledge: Theory, Hypothesis, Claim, Equation, Definition, Finding, Reference
Research: Paper, Book, Chapter, Experiment, Benchmark, Dataset, Model
Agentic: Agent, Skill, HyperSkill, Ability, HyperAbility, HyperLoop, Workflow, Tool
Software: Repository, Package, Service, API, Component, Config, Deployment
Governance: Decision, Policy, Constraint, Risk, Audit, Incident, Migration, Exception
Operations: Goal, Task, Plan, Execution, Result, Failure, Recovery, Checkpoint
```

العلاقات المعيارية

```
IS | HAS | PART_OF | BELONGS_TO
DERIVED_FROM | BASED_ON | REFERENCES | SUPPORTED_BY
VERSION_OF | SUPERSEDES | REPLACES | DEPRECATED_BY
EXTENDS | SPECIALIZES | GENERALIZES | IMPLEMENTS
TRANSLATION_OF | SUMMARY_OF | VIEW_OF | EXPORT_OF
FORK_OF | MERGED_INTO
CONTRADICTS | COMPETES_WITH | VALIDATES | FALSIFIES
USES | PRODUCES | CONSUMES | GOVERNS
OWNED_BY | CREATED_BY | MODIFIED_BY | APPROVED_BY
```

الفصل الابستيمي

كل claim يجب أن يحمل `epistemic_type` من:

```
FACT | CLAIM | HYPOTHESIS | THEORY | ASSUMPTION |
INFERENCE | DESIGN_DECISION | SPECULATION
```

لا يسمح للmerge بتحويل نوع إبستيمي إلى آخر دون سجل قرار صريح.

البعد الزمني

```
valid_from:
valid_until:
observed_at:
superseded_at:
```

الحقيقة المتغيرة زمنياً تحفظ كتاريخ حالات، لا overwrite.

ملحق: SCHEMAS 05

Schema Object Canonical

```yaml
mcarg_id: MCARG:<TYPE>:<NAMESPACE>:<NNNN>
canonical_name: string
entity_type: enum
scope: string
status: active|draft|superseded|archived|deprecated|quarantined
canonical_uri: string|null
aliases: []
version: semver-or-monotonic
parent_version: string|null
supersedes: []
created_at: timestamp
updated_at: timestamp
created_by: actor_id
modified_by: actor_id
source_ids: []
derived_from: []
relations: []
content_hash: sha256
semantic_fingerprint: string
structural_fingerprint: string
confidence: 0.0..1.0
verification_state: unknown|checked|verified|reproduced
canonicality_score: 0.0..1.0
duplicate_risk_score: 0.0..1.0
writeback_status: pending|committed|rolled_back
integrity_status: healthy|warning|broken|quarantined
provenance: {}
lineage: {}
```

Record Registry

```json
{
  "mcarg_id": "MCARG:THEORY:MCD:0001",
  "canonical_pointer": "canonical/theories/MCD/current",
  "active_version": "3.2.0",
  "scope": "research-main",
  "authority": "canonical",
```

```
  "aliases": ["MCD", "Multi-Context Dynamics"],
  "relations": [],
  "last_verified_at": "<timestamp>",
  "integrity": "healthy"
}
```

record Event

```
{
  "event_id": "evt-...",
  "actor": "agent-or-human",
  "action": "VERSION",
  "target_id": "MCARG:...",
  "parent_version": "3.1.0",
  "new_version": "3.2.0",
  "reason": "new evidence",
  "source_ids": ["SRC:..."],
  "timestamp": "...",
  "transaction_id": "tx-..."
}
```

ملحق: PROTOCOL ID CANONICAL 06

الهدف

تثبيت الهوية بحيث لا يتغير معنى الكيان بتغير الاسم أو المجلد أو صيغة الملف.

الصيغة

```
MCARG:<ENTITY_TYPE>:<NAMESPACE>:<SEQUENCE>
```

أمثلة:

```
MCARG:THEORY:MCD:0001
MCARG:PAPER:AGENTIC-ARCH:0042
MCARG:SKILL:DEDUP:0177
MCARG:AGENT:LINEAGE-AUDITOR:0021
```

قواعد

1. المعرف immutable بعد الإصدار.
2. لا يعاد استخدام معرف retired لكيان جديد.
3. تغيّر الاسم يسجل alias أو rename event.

4. split ينتج هويات جديدة مرتبطة بـDERIVED_FROM.
5. merge يحافظ على IDs القديمة كـMERGED_INTO ولا يمحوها.
6. scope جزء من canonicality وليس بالضرورة جزءاً من ID.

Resolution Identity

```
normalize request
→ collect exact-id matches
→ alias lookup
→ lexical candidates
→ semantic candidates
→ graph candidates
→ provenance candidates
→ classify
→ resolve or quarantine
```

حالات عدم اليقين

إذا لم يتجاوز أفضل مرشح policy threshold أو كان المرشحان متقاربان، يجب أن تكون النتيجة UNRESOLVED_IDENTITY لا اختياراً عشوائياً.

ملحق: INVARIANT DUPLICATION ZERO 07

Invariant Zero-Duplication — ZDI

التعريف الرسمي داخل نطاق s وزمن t:

$$\forall x, y \in ActiveCanonical(s,t) : Identity(x) = Identity(y) \Rightarrow x = y$$

ما لا يعنيه ZDI

ZDI لا يمنع:

- نسخ backup.
- PDF صادر من Markdown.
- ترجمة عربية/إنجليزية.
- versions، historical
- branches صريحة.
- cache أو embeddings.
- views، materialized

بل يشترط أن تكون علاقاتها وهويتها المرجعية معلنة.

فئات الازدواج

```
D0 Binary duplicate
D1 Formatting duplicate
D2 Semantic duplicate
D3 Version duplicate
D4 Partial overlap
D5 Translation duplicate
D6 Derived artifact
D7 Fork
D8 False duplicate
```

قاعدة الأمان

أي عملية dedup غير قادرة على إثبات أن المعلومة الفريدة محفوظة يجب أن تفضل LINK/QUARANTINE على DELETE/MERGE.

ملحق: ENGINE RESOLUTION DUPLICATE 08

Engine Resolution Duplicate — DRE

مرحلة generation candidate

استعمل عدة فهارس بالتوازي:

```
exact hash index
normalized hash index
name/alias index
semantic vector index
structure fingerprint index
provenance index
graph neighborhood index
```

التصنيف

النتيجة ليست true/false duplicate: فقط، بل:

```yaml
relationship: EXACT_DUPLICATE|FORMAT_VIEW|SEMANTIC_DUPLICATE|VERSION|
              PARTIAL_OVERLAP|TRANSLATION|DERIVED|FORK|DISTINCT|UNRESOLVED
confidence: 0..1
evidence:
  hash: ...
  semantic: ...
  graph: ...
```

```
  provenance: ...
recommended_action: ...
```

thresholds افتراضية تجريبية

```
>= .98 exact/probable exact
.92-.98 probable semantic duplicate
.82-.92 possible version or strong overlap
.65-.82 related
< .65 usually distinct
```

هذه الحدود تحتاج calibration على corpus الحقيقي.

contradiction قاعدة

إذا كان الموضوع واحداً لكن propositions غير متوافقة، أعد `CONTRADICTION` ولا تستخدم `SEMANTIC_DUPLICATE`.

ملحق: 09 PROTOCOL ARTIFACT CANONICAL

Protocol Artfiact Canonical — CAP

CAP هو بوابة artfiact-level داخل MCARG.

البوابات السبع

```
CAP-1 Identity
CAP-2 Existing-State Search
CAP-3 Difference Extraction
CAP-4 Relationship Classification
CAP-5 Provenance Attachment
CAP-6 Canonical Commit
CAP-7 Integrity Verification
```

pseudo-flow

```
def canonical_artifact_write(proposal):
    identity = resolve_identity(proposal)
    existing = retrieve(identity)
    delta = semantic_diff(existing, proposal)
    relation = classify(existing, proposal)
    action = policy_select(relation, delta)
    tx = begin_transaction()
    apply(action, preserve_provenance=True)
    update_registries()
```

```
    update_graph()
    verify_integrity()
    commit(tx)
```

مخرجات CAP

كل استدعاء يجب أن يعيد:

```
canonical_id
action_taken
version_before/version_after
relationships_written
provenance_written
integrity_result
transaction_id
```

ملحق: GRAPH LINEAGE PROVENANCE 10

Graph Lineage & Provenance — PLG

الـprovenance يجيب: **من أين جاءت هذه المعلومة؟** والـlineage يجيب: **كيف تحولت حتى وصلت إلى حالتها الحالية؟**

عقدة lineage

```yaml
node_id:
mcarg_id:
version:
parents: []
operation: create|update|merge|translate|derive|retract
actor:
sources: []
reason:
timestamp:
content_hash:
```

خواص واجبة

- traceability من أي نسخة إلى مصادرها.
- لا merge بلا parent set.
- لا claim source بلا record SOURCE_REGISTRY حين يكون المصدر معروفاً.
- append-only lineage historical منطقياً.
- التصحيح لا يمحو الخطأ؛ بل يضيف correction/supersede edge.

العلاقات

```
DERIVED_FROM
VERSION_OF
TRANSLATION_OF
SUMMARY_OF
SUPPORTED_BY
CONTRADICTS
VALIDATES
FALSIFIES
SUPERSEDES
```

reconstruction

من PLG وVERSION_REGISTRY يجب أن يمكن إعادة تكوين تسلسل التغييرات والتحقق من أن pointer active متوافق مع أحدث commit صالح.

ملحق: PROTOCOL MERGE VERSION 11

Protocol Merge & Version — VMP

rule version

التغيير الدلالي يجب أن ينتج version أو event واضح. التغيير الصرفي في rendering يمكن أن ينتج export جديداً دون رفع النسخة الدلالية، وفق policy.

merge three-way semantic

ليكن C الجد المشترك، وA,B فرعين:

$$M = Merge(C, A, B)$$

صنّف الوحدات:

```
UNCHANGED
ADDED_A
ADDED_B
REMOVED_A
REMOVED_B
CONFLICT
CONTRADICTION
SUPERSEDED
```

قواعد

- الدمج لا يساوي concatenation.
- provenance لكل claim يحتفظ بمصدره.
- contradiction تحفظ كعلاقة.
- إذا فشل الدمج الآلي في الحفاظ على unique information، أوقف commit.
- canonicality branch scoped، يمكن أن توجد branches production/research/experimental دون خرق ZDI.

concurrency optimistic

```
read version 17
commit only if current_version == 17
else MCARG_CONFLICT
```

ثم rebase/merge، لا overwrite.

ملحق: PROTOCOL COMMIT REGISTRY 12

Protocol Commit Registry — RCP

transaction

```
BEGIN
RESOLVE IDENTITY
LOCK/CHECK EXPECTED VERSION
CALCULATE DIFF
VALIDATE POLICY
WRITE ARTIFACT
WRITE PROVENANCE
WRITE VERSION EVENT
WRITE REGISTRIES
WRITE GRAPH
VERIFY HASHES/REFERENCES
COMMIT
```

عند الفشل:

```
ROLLBACK
or
QUARANTINE
```

objective atomicity

لا يجوز أن تكون لدينا حالة مثل:

```
artifact updated = true
registry updated = false
```

من دون recovery، marker. إذا كان backend لا يدعم transaction حقيقية، استعمل write-ahead log + marker، commit + replay idempotent

idempotency

كل operation world-effect يملك `idempotency_key` حتى لا يعيد Agent تنفيذ commit نفسه بعد timeout أو retry.

ملحق: 13 RECONSTRUCTION SELF

Sefl-Reconstruction Integrity — ISR

الهدف

استرجاع النظام بعد context loss، index corruption، model swap partial، storage failure، أو migration.

sequence boot

```
LOAD MASTER REGISTRIES
→ VERIFY SCHEMAS
→ VERIFY CONTENT HASHES
→ LOAD CANONICAL IDS
→ REBUILD VERSION POINTERS
→ REBUILD RELATIONSHIP GRAPH
→ REBUILD SEARCH/VECTOR INDEXES
→ FIND ORPHANS
→ DETECT DUPLICATES
→ RESTORE ACTIVE GOALS
→ RESTORE AGENTS/SKILLS/LOOPS
→ RUN INTEGRITY AUDIT
→ RESUME
```

مصادر الحقيقة

الأولوية المقترحة:

```
1. immutable event/commit log
2. canonical registries
3. provenance/lineage graph
4. canonical content store
5. derived indexes/caches
```

الـindex vector ليس source of truth، يمكن إعادة بنائه.

invariant recovery

$$Similarity(State_{reconstructed}, State_{last_valid}) \to 1$$

مع تقرير صريح لأي عناصر لم يمكن استعادتها.

ملحق: PACK AGENT 14

Pack Agent MCARG

ID	Agent	الوظيفة
A01	Canonical Registrar	إنشاء/حل canonical IDs
A02	Duplicate Hunter	البحث عن الازدواج الصريح والدلالي
A03	Semantic Merge Judge	merge/version/fork/distinct
A04	Provenance Keeper	حفظ الأدلة والنسب
A05	Version Controller	إدارة التطور التاريخي
A06	Graph Curator	إصلاح العلاقات
A07	Orphan Hunter	كشف artfiacts غير المسجلة
A08	Conflict Arbiter	حل concurrent mutations
A09	Integrity Auditor	تدقيق عالمي
A10	Reconstruction Agent	إعادة بناء الحالة
A11	Canonicality Judge	اختيار المرجع سلطة الأعلى
A12	Migration Agent	النقل أثناء الهوية على الحفاظ
A13	Archive Curator	أرشفة superseded state
A14	Policy Guardian	الحوكمة قوانين تطبيق
A15	Writeback Controller	gate لكل world effect

duties of separation

يفضل ألا يمتلك agent واحد صلاحية `propose + approve + commit + audit` في التغييرات عالية الخطورة. يمكن جمع الأدوار في التطوير المحلي، لكن policy production تفصلها منطقياً.

contract message

```
proposal_id:
actor_agent:
target_identity:
```

```yaml
requested_action:
evidence: []
expected_version:
risk_class:
policy_context:
```

ملحق: PACK HYPERSKILL 15

S001-S060 HyperSkills

```
S001 ResolveCanonicalIdentity
S002 GenerateCanonicalID
S003 NormalizeArtifact
S004 ComputeContentHash
S005 ComputeSemanticFingerprint
S006 ComputeStructuralFingerprint
S007 FindDuplicateCandidates
S008 ClassifyDuplicate
S009 DetectVersionRelationship
S010 DetectTranslationRelationship
S011 DetectFork
S012 DetectContradiction
S013 CalculateSemanticDiff
S014 ThreeWaySemanticMerge
S015 PreserveProvenance
S016 BuildLineageEvent
S017 RegisterEntity
S018 RegisterArtifact
S019 RegisterSource
S020 RegisterVersion
S021 UpdateLatestState
S022 UpdateRelationshipGraph
S023 ResolveAlias
S024 DetectOrphan
S025 RepairBrokenReference
S026 ValidateCanonicalUniqueness
S027 AuditRegistry
S028 AuditGraph
S029 AuditLineage
S030 RebuildSearchIndex
```

S031 RebuildVectorIndex
S032 ReconstructState
S033 QuarantineConflict
S034 GenerateDuplicateReport
S035 GenerateIntegrityReport
S036 ArchiveSupersededState
S037 VerifyWriteback
S038 RollbackTransaction
S039 ReplayTransaction
S040 CheckIdempotency
S041 ValidateExpectedVersion
S042 CalculateCanonicalityScore
S043 CompareSourceAuthority
S044 DetectTemporalConflict
S045 ScopeCanonicality
S046 LinkDerivedArtifact
S047 RegisterTranslation
S048 RegisterExport
S049 RegisterSummary
S050 PreserveEpistemicType
S051 DetectInformationLoss
S052 ValidateMergeCompleteness
S053 CheckSchemaCompatibility
S054 MigrateCanonicalPointer
S055 ReconcileRegistryAndStore
S056 ReconcileGraphAndRegistry
S057 RestoreCheckpoint
S058 VerifyCheckpoint
S059 GenerateAuditTrail
S060 EnforceWorldEffectGate

contract skill

كل HyperSkill يجب أن يعلن `inputs`, `outputs`, `preconditions`, `side_effects`, `idempotency`, `failure_modes`, و`evidence_required`.

ملحق: PACK HYPERLOOP 16

HyperLoops

Deduplication Continuous L01

```
LEARN → VERIFY → REPAIR → CLASSIFY → COMPARE → RETRIEVE → SCAN
```

Integrity Registry L02

```
RECONCILE → COMPARE → INDEX → GRAPH → STORE → REGISTRY
```

Sefl-Reconstruction L03

```
VERIFY → REBUILD STATE → REBUILD GRAPH → LOAD CANONICAL → FAILURE → CHECKPOINT
```

Consolidation Knowledge L04

```
CANONICALIZE → MERGE → VALIDATE → DELTA → EXISTING → KNOWLEDGE NEW
```

Verfiication Provenance L05

```
UPDATE → CONFIDENCE → RELIABILITY → EVIDENCE → SOURCE → CLAIM
```

Evolution Capability L06

```
PROMOTE/REJECT → EVAL → CANDIDATE → DELTA → SKILL EXISTING → WEAKNESS
```

Maintenance Truth Temporal L07

```
VERIFY → SUPERSEDE/APPEND → TIME COMPARE → ENTITY RESOLVE → OBSERVE
```

Recovery Orphan L08

```
LINK/QUARANTINE → VERIFY → ID CANDIDATE INFER → ORPHAN → SCAN
```

termination

كل loop يحتاج stop criteria، maximum budget، escalation retries، rule وcheckpoint؛ وإلا يصبح sefl-maintenance نفسه مصدراً لعدم الاستقرار.

ملحق: BENCHMARK EVALUATION 17

MCARG-Bench

مجموعات المهام

1. Exact duplicate detection.

2. detection. duplicate Near/semantic
3. recognition. Version
4. recognition. Translation
5. detection. Fork
6. preservation. Contradiction
7. resolution. identity Canonical
8. merge. three-way Semantic
9. retention. Provenance
10. recovery. Orphan
11. loss. context after Reconstruction
12. conflicts. write Concurrent
13. integrity. Registry
14. preservation. Information
15. updates. truth Temporal
16. identity. Cross-format
17. resistance. duplicate False-positive
18. collisions. alias Adversarial

metrics

$$DR = \frac{UncontrolledDuplicates}{GovernedEntities}$$

$$CC = \frac{CanonicalizedEntities}{TotalGovernedEntities}$$

$$PC = \frac{ClaimsWithProvenance}{Claims}$$

$$OR = \frac{OrphanArtifacts}{Artifacts}$$

$$RF = Similarity(State_{reconstructed}, State_{canonical})$$

$$IP = \frac{UniqueInformationAfterMerge}{UniqueInformationBeforeMerge}$$

design benchmark

يجب فصل corpora dev/test حسب المشاريع والأنواع لتفادي تسرب alias، patterns وقياس precision/recall لكل class لا accuracy aggregate فقط.

ملحق: PACK TRAINING 18

تدريب الهوية والحوكمة

trajectories SFT

أمثلة :supervised

```
duplicate → ALIAS/REFERENCE
revision → VERSION
contradiction → PRESERVE_BOTH + CONTRADICTS
translation → TRANSLATION_OF
new concept → CREATE
ambiguous → QUARANTINE/REQUEST_EVIDENCE
```

training contrastive

كوّن أزواجاً صعبة:

```
same concept vs closely related concept
version vs independent artifact
translation vs inspired-by artifact
fork vs update
summary vs replacement
contradiction vs semantic duplicate
```

perturbation synthetic

ابدأ من artfiact مرجعي وطبّق:

```
rename, translate, summarize, expand, reorder, paraphrase,
split, merge, slightly update, contradict, fork, reformat
```

ثم درّب classfiier/controller على العلاقة الصحيحة.

objective preference

كافئ: preservation، information linking، correct duplication، minimal provenance، reuse،

عاقب: silent creation، canonical duplicate lineage، lost merge، destructive state، fabricated overwrite،

curriculum tool-use

```
BEFORE VERIFY → REGENERATE BEFORE DIFF → CREATE BEFORE RESOLVE → WRITE BEFORE SEARCH
.COMMIT
```

ملحق: RECOVERY FAILURE 19

Taxonomy Failure

```
F01 Duplicate Canonical Identity
F02 Missing Canonical ID
F03 Broken Provenance
F04 Broken Parent Version
F05 Orphan Artifact
F06 Cyclic Version Graph
F07 Conflicting Authority
F08 Stale Latest-State Pointer
F09 Missing Source
F10 Invalid Alias
F11 Semantic Merge Loss
F12 Silent Overwrite
F13 Concurrent Mutation
F14 Index/Registry Divergence
F15 Graph/Store Divergence
F16 Hallucinated Artifact Reference
F17 Unauthorized Mutation
F18 Corrupted Content
F19 Unresolved Contradiction
F20 Reconstruction Failure
```

pipeline repair

```
DETECT
→ ISOLATE
→ DIAGNOSE
→ FIND LAST VALID STATE
→ RECONSTRUCT
→ REPLAY VALID CHANGES
→ VERIFY
→ WRITE AUDIT RECORD
```

quarantine-first

في حالات الشك لا تحذف الأصل. انقل منطقياً إلى حالة `QUARANTINED` مع سبب، ثم اسمح للمراجع أو agent أعلى سلطة بحسمها.

ملحق: GOVERNANCE SECURITY 20

Model Security

MCARG يحمي integrity لا confidentiality فقط. التهديدات تشمل:

- poisoning عبر artfiacts متشابهة الأسماء.
- hijacking، alias
- version، stale to rollback
- provenance، fabricated
- merge، malicious
- commit، prior of replay
- overwrite، concurrent
- escalation privilege عبر agent، tool-use

controls

```
RBAC/ABAC per action
signed or hashed commit events
append-only audit log
expected-version checks
idempotency keys
source verification
quarantine for ambiguity
separation of duties
rate/budget limits
policy-as-code
```

gate world-effect

لا يكتب Agent مباشرة إلى canonical، storage المسار:

```
Worker → Resolver → Policy Guardian → Writeback Controller → Store → Auditor
```

actions destructive

DELETE, high-impact MERGE, mass MOVE, authority، change وmigration schema تحتاج policy أعلى، وقد تتطلب approval human حسب risk، class

ملحق 21: API RUNTIME

API Runtime

```
resolve_identity(obj, scope=None)
find_candidates(obj, top_k=20)
compare(a, b)
classify_relationship(a, b)
semantic_diff(a, b)
create_canonical(obj, scope)
register_alias(alias, canonical_id)
create_version(canonical_id, delta, expected_version)
merge_versions(a, b, ancestor=None)
attach_provenance(target, sources)
register_relationship(a, relation, b)
begin_transaction()
commit(transaction)
rollback(transaction)
audit_registry()
audit_duplicates()
audit_lineage()
reconstruct_state(checkpoint=None)
```

CLI

```
mcarg resolve <entity>
mcarg diff <A> <B>
mcarg canonicalize <artifact>
mcarg merge <A> <B>
mcarg link <A> <relation> <B>
mcarg audit duplicates
mcarg audit lineage
mcarg audit global
mcarg repair
mcarg reconstruct
mcarg checkpoint
mcarg status
```

envelope response

```
{
  "ok": true,
  "transaction_id": "tx-...",
  "canonical_id": "MCARG:...",
  "action": "VERSION",
  "version_before": "1.2.0",
  "version_after": "1.3.0",
  "evidence": [],
  "integrity": "healthy"
}
```

SUITE TEST 22 :ملحق

Strategy Test

Unit

- uniqueness. ID
- resolution. alias
- normalization. hash
- boundaries. classfiier relationship
- updates. pointer version
- append. provenance
- idempotency.

Property-based

```
merge never removes unique evidence
re-running same transaction has no extra side effect
active canonical count per scope <= 1
all active objects resolve to existing registry records
all parent versions are reachable
```

Integration

- commits. concurrent multi-agent
- update. registry and write artfiact between crash
- reconstruction. then loss index
- entity. same of export + summary + translation
- claim. same on papers contradictory

Chaos

- mid-commit، process kill
- index، vector corrupt
- cache، remove
- snapshot، registry duplicate
- replay، event reorder

targets Acceptance

`CanonicalCoverage` target: prototype research Example risk، production from set be should Targets explicit with ,`.995 >= ReconstructionFidelity` ,`.005 <= UncontrolledDuplicateRate` ,`.99 >=` sets، benchmark over intervals confidence

IMPLEMENTATION REFERENCE 23 :ملحق

Architecture Reference Minimal

```
API Gateway
  ↓
Identity Resolver
  ↓
Candidate Retrieval ──→ Vector/Search Index
  ↓
Relationship Classifier
  ↓
Policy Engine
  ↓
Transaction Manager
  ├─ Canonical Object Store
  ├─ Registry DB
  ├─ Lineage/Event Log
  └─ Relationship Graph
  ↓
Integrity Auditor
```

contract storage technology-neutral

- store، relational or key/value transactional **DB: Registry**
- documents، versioned or blobs content-addressed immutable **Store: Object**
- stream/table، append-only **Log: Event**
- table، edge relational or DB graph **Graph:**

index، vector rebuildable **Index: Semantic** •

pseudo-code

```
def governed_write(proposal, scope):
    norm = normalize(proposal)
    cand = find_candidates(norm)
    res = resolve(norm, cand, scope)
    action = choose_action(res)
    tx = begin()
    try:
        result = apply(action, tx, preserve_provenance=True)
        validate_zdi(scope, tx)
        validate_lineage(tx)
        validate_information_preservation(tx)
        commit(tx)
        return result
    except Exception:
        rollback(tx)
        raise
```

ARCHITECTURE DEPLOYMENT 24 :ملحق

Profiles Deployment

single-user / Local

module يكون أن يمكن service كل index، vector local + store filesystem/object + SQLite/PostgreSQL داخل process واحد مع log، transaction

lab research / Team

policy centralized workers، audit async layer، graph versioned، storage object registry، PostgreSQL engine.

runtime AGI Distributed

```
Agent Mesh
→ MCARG Gateway
→ Identity/Policy Cluster
→ Transaction Service
→ Registry Consensus Store
→ Object Store
```

→ Event/Lineage Stream

→ Graph + Search Indexes

→ Audit/Reconstruction Workers

consistency

pointer Canonical وregistry version يحتاجان consistency strong أو Search/vector compare-and-swap. indexes يمكن أن تكون consistent eventually لأنها مشتقة وليست مصدر حقيقة.

policy availability

عند تعذر canonical، registry يفضل النظام `mode degraded read-only` على writes عمياء قد تولد ازدواجاً.

ملحق: OBSERVABILITY METRICS 25

Metrics

```
mcarg_canonical_coverage
mcarg_duplicate_rate
mcarg_orphan_rate
mcarg_provenance_coverage
mcarg_merge_information_preservation
mcarg_reconstruction_fidelity
mcarg_identity_resolution_latency
mcarg_commit_latency
mcarg_conflict_rate
mcarg_quarantine_rate
mcarg_stale_pointer_count
mcarg_broken_lineage_edges
```

Health Global

مثال قابل للتعديل:

$$G = 0.25CC + 0.20PC + 0.15RF + 0.15IP + 0.15(1 - DR) + 0.10(1 - OR)$$

events observability

كل decision عالي الأثر يجب أن يسجل: candidate set، policy scores، selected version، action، reasons، evidence، latency، actor وtransaction. result.

dashboards

- overview، integrity
- review، awaiting clusters duplicate
- queue، identity unresolved
- breaks، lineage
- readiness، reconstruction
- thresholds، classfiier of drfit

ملحق: PLAYBOOK MIGRATION 26

MCARG to Migration

cleanup destructive Freeze — 0 Phase

لا تبدأ بحذف المكررات. أنشئ read-only inventory أولاً.

Inventory — 1 Phase

استخراج الملفات، artfiacts، source registries، links، hashes، metadata، timestamps، owners.

Fingerprinting — 2 Phase

احسب fingerprints، semantic/structural + hashes exact/normalized

clusters Candidate — 3 Phase

كوّن clusters محتملة دون merge.

assignment Canonical — 4 Phase

حدد canonical IDs، scope، aliases، derived versions، views، contradictions.

bacfkill Provenance — 5 Phase

استخرج source/parent/creator حيث يمكن، وضع `unknown` صراحة حين لا يمكن.

build Registry — 6 Phase

ابنِ registries، ENTITY/ARTIFACT/SOURCE/VERSION/LATEST_STATE

build Graph — 7 Phase

اكتب edges، lineage/relationship

Validation — 8 Phase

review، sample random audit، broken-edge audit، orphan audit، ZDI شغل

Cutover — 9 Phase

اجعل كل writes الجديدة تمر عبر CAP/RCP.

duplicates legacy Archive — 10 Phase

بعد validation فقط، صنف copies legacy كـarchive/alias/export؛ لا تدمر provenance.