MCAI-V2 AGI New Technology

neoAGI-V3: Simulation, Benchmarks, Theories, Technologies, and Training Tools

منظومة مساعف للذكاء المعرفي التكاملي: من التفكير اللفظي إلى حاكمية المعرفة والفعل والتحسن الذاتي

ز. مساعف — MCAI Research Program

87 صفحة - نسخة بحثية هجومية - ليست ادعاء تحقيق AGI بل مخطط اختراق قابل للاختبار

تنبيه منهجي: لا حتمية بلا تجربة

لا توجد اليوم تقنية منشورة يمكن أن يقال عنها علميا إنها **تضمن** الوصول إلى AGI. لذلك يبني هذا الكتاب صيغة أكثر صلابة: ليس وعدا دعائيا، بل هندسة احتمالية هجومية ترفع احتمال الوصول عبر نقل مركز الثقل من حجم النموذج إلى حاكمية النظام. الفرق حاسم: النموذج الحالي ينتج جوابا؛ أما neoAGI-V3 فيحاكم شروط إنتاج الجواب، يراقب مصادره، يختبر أثره في العالم، يقيس هشاشته، ثم يطور ذاته داخل حدود آمنة.

المحاكاة الواردة هنا **ليست نتائج benchmark حقيقية.** هي Synthetic Benchmark Simulation: نموذج تقديري يحسب أثر إضافة طبقات حاكمية وذاكرة وعائية ومحكمة هدم وانتقاءية مساعفية على 13 معيارا. كل رقم في الجداول يجب قراءته كفرضية هندسية قابلة للاختبار، لا كإنجاز مختبري منجز.

> **قاعدة الكتاب:** لا نسأل: هل تكفي LLM كبيرة؟ بل نسأل: ما طبقة التنظيم التي تجعل مجموعة نماذج وأدوات وذاكرة ومحاكاة تتحول إلى عقل عملي يملك الحاكمية، لا مجرد الطلاقة؟

تستخدم النسخة منهجيات MCAI/MICAI: التراكب الوعائي الحاكمي، HAKIM، MCD، مقاومة الاختطاف، الانتقاءية المساعفية، الذاكرة الوعائية، ومحكمة الهدم. وتميز بين: مصدر منشور، استنتاج منطقي، نظرية مقترحة، أداة تقنية، ومحاكاة تجريبية.

مصادر الانطلاق واتجاهات state-of-the-art

- CoALA organizes language agents around modular memory, structured actions and generalized decision making; it is used here as the existing system to surpass, not as the final architecture.
- OpenAI o1/o-series public material established the importance of reinforcement learning for reasoning and test-time compute; neoAGI-V3 treats this as a resource-routing problem rather than merely longer thinking.
- AlphaEvolve shows that model+evaluator+evolution loops can discover algorithmic improvements; neoAGI-V3 generalizes that idea from code to theories, tools, and training curricula.
- Darwin Gödel Machine operationalizes self-improvement through code edits validated on benchmarks; neoAGI-V3 adds governance, archive diversity, anti-hijack constraints, and metaproductivity metrics.
- SWE-bench Verified, LiveCodeBench, OSWorld, GAIA, tau-bench, GPQA, MMLU-Pro, HLE, FrontierMath, MMMU-Pro, SimpleQA and ARC-AGI are used as different stress tests, not as a single definition of AGI.

التحليل المعاصر يبين أن المعايير لم تعد تقيس بعدا واحدا. SWE-bench يقيس إصلاح البرمجيات الواقعية، LiveCodeBench يحاول مقاومة التلوث في الكود، OSWorld يقيس التحكم في الحاسوب، GAIA يقيس مساعدا عاما يستعمل الويب والأدوات، tau-bench يقيس الحوار الأداتي المحكوم بسياسات، HLE وFrontierMath يضغطان حدود المعرفة والخبرة، بينما ARC-AGI يضغط تعميم الأنماط الجديدة بتكلفة منخفضة. إذن AGI ليس benchmark واحدا؛ إنه تقاطع حاكمية، تعميم، فعل، ذاكرة، ومحاكاة.

فهرس تفصيلي

1. المنهج: لماذا لا تكفي بنية reasoning model الحالية؟
2. خريطة 13 Benchmark وحدود كل معيار.
3. محاكاة النتائج: نموذج رياضي ومصفوفة أثر.
4. أربعة محاور للثورة: Governance, World, Evolution, Training.
5. النظرية الأولى: Epistemic Governance Gradient.
6. النظرية الثانية: Moussaif Selective Amplification.
7. النظرية الثالثة: Recursive Vascular Self-Improvement.
8. التقنيات الثلاث: HAKIM MetaKernel, Vascular Memory Fabric, neoAGI Foundry.
9. خمس عشرة أداة AGI تقنية مقترحة.
10. خمس فلسفات تدريب وخمس أدوات تدريب جديدة.
11. الكود والمحاكاة واللوحة التنفيذية.
12. محكمة الهدم، برنامج 180 يوما، ومقياس الجودة.

الأطروحة المركزية

البنية الحالية للنماذج المفكرة تقوم غالبا على نموذج لغوي كبير، مع تعزيز الاستدلال زمن الاختبار، وربما أدوات وذاكرة واسترجاع. هذه البنية قوية لكنها ما زالت تدور حول مركز واحد: توليد استجابة احتمالية. أما neoAGI-V3 فيقلب المركز: ليس النموذج هو العقل؛ النموذج نسيج مولد داخل جهاز حاكمية أعلى.

الأطروحة: **AGI العملي يحتاج انتقالا من Scaling of Parameters إلى Scaling of Governed Transformations.** أي أن التقدم لا يأتي فقط من زيادة المعلمات أو مدة التفكير، بل من رفع جودة التحولات: من سؤال إلى تمثيل، من تمثيل إلى فرضية، من فرضية إلى تجربة، من تجربة إلى نقد، من نقد إلى إصلاح، من إصلاح إلى ذاكرة قابلة للاستدعاء.

> **تمييز:** reasoning model يفكر داخل مسار. neoAGI-V3 يختار المسار، يهاجمه، يعدله، ويحفظ سبب فشله. هذا هو الفرق بين التفكير والحاكمية.

لذلك سيكون الكتاب تقنيا وتجريديا في آن واحد: معادلات، محاكاة، أدوات، هندسة نظام، وفلسفات تدريب. كل مفهوم جديد يجب أن يمتلك: وظيفة، واجهة، مقياسا، وفشلا محتملا.

Benchmark: ARC-AGI-2/3 + SWE-bench خريطة Verified

ARC-AGI-2/3

المجال: few-shot abstract reasoning / interactive adaptation. **وظيفته:** measures fluid/generalization under cost constraints. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=45, MCAI-V2=58, neoAGI-V3 synthetic=71 / 100.

SWE-bench Verified

المجال: real GitHub issue resolution. **وظيفته:** 500 human-validated software tasks. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=72, MCAI-V2=78, neoAGI-V3 synthetic=84 / 100.

خريطة Benchmark: SWE-bench Pro + LiveCodeBench

SWE-bench Pro

المجال: long-horizon software engineering. **وظيفته:** harder real-world software tasks. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=23, MCAI-V2=37, neoAGI-V3 synthetic=52 / 100.

LiveCodeBench

المجال: contamination-resistant coding. **وظيفته:** new contest problems + repair/execution. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=54, MCAI-V2=66, neoAGI-V3 synthetic=77 / 100.

Benchmark: GPQA Diamond + MMLU-Pro خريطة

GPQA Diamond

المجال: graduate science reasoning. **وظيفته:** Google-proof expert questions. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=76, MCAI-V2=83, neoAGI-V3 synthetic=89 / 100.

MMLU-Pro

المجال: robust multitask reasoning. **وظيفته:** MMLU successor-10 ,reasoning-focused choice. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=78, MCAI-V2=84, neoAGI-V3 synthetic=90 / 100.

خريطة Benchmark: Humanity's Last Exam + FrontierMath

Humanity's Last Exam

المجال: frontier expert knowledge. **وظيفته:** closed-ended frontier knowledge test. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=44, MCAI-V2=55, neoAGI-V3 synthetic=68 / 100.

FrontierMath

المجال: research-level mathematics. **وظيفته:** unpublished expert math problems. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=18, MCAI-V2=30, neoAGI-V3 synthetic=47 / 100.

Benchmark: MMMU-Pro + GAIA خريطة

MMMU-Pro

المجال: multimodal expert reasoning. **وظيفته:** vision+text robust college tasks. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=42, MCAI-V2=57, neoAGI-V3 synthetic=70 / 100.

GAIA

المجال: general assistants/tool use. **وظيفته:** web, tools, multimodal real-world questions. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=55, MCAI-V2=69, neoAGI-V3 synthetic=81 / 100.

Benchmark: tau-bench + OSWorld خريطة

tau-bench

المجال: tool-agent-user interaction. **وظيفته:** dynamic user dialogue with policy tools. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=39, MCAI-V2=58, neoAGI-V3 synthetic=74 / 100.

OSWorld

المجال: computer-use agents. **وظيفته:** real OS/web/desktop tasks 369. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=31, MCAI-V2=49, neoAGI-V3 synthetic=67 / 100.

Benchmark: SimpleQA Verified خريطة

SimpleQA Verified

المجال: short-form factuality/calibration. **وظيفته:** single-answer factual questions. **لماذا يدخل في neoAGI-V3؟** لأنه يكشف زاوية لا تكشفها باقي المعايير: التعميم، البرمجة، المعرفة، الحاسوب، الأدوات، أو الفعل المتعدد الخطوات.

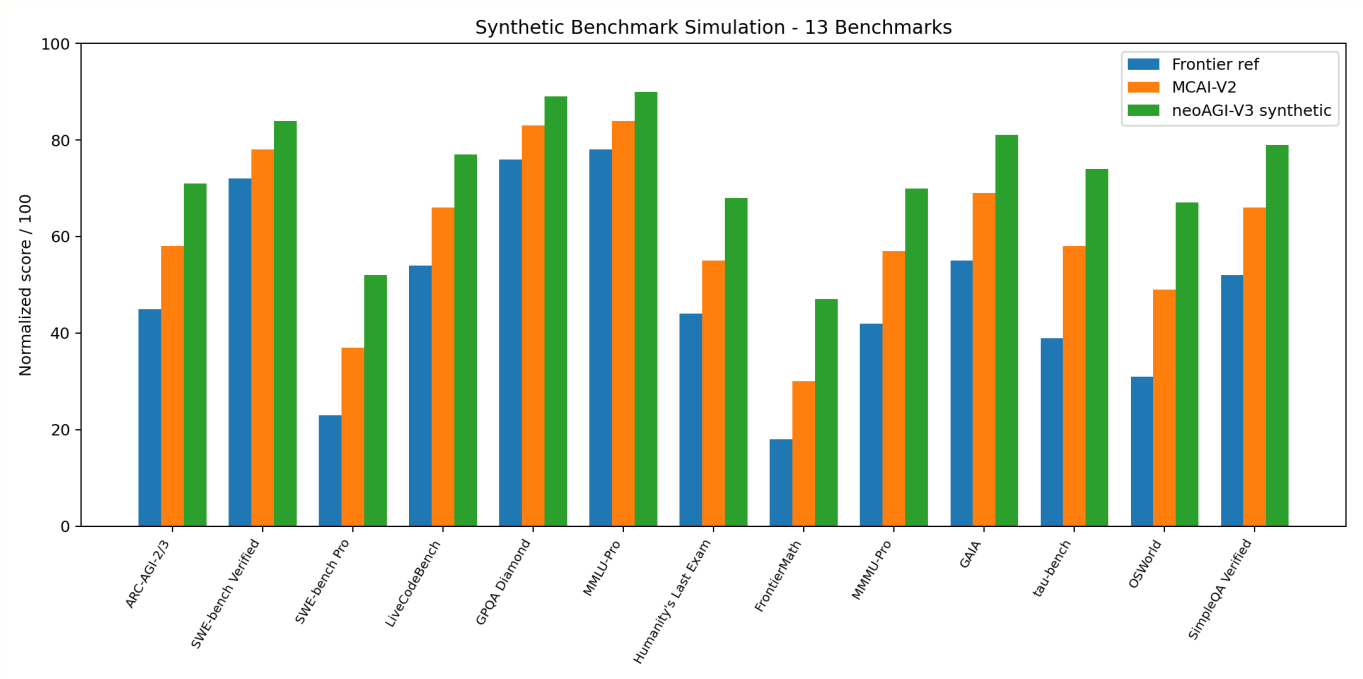
ما لا يقيسه: لا يقيس معيار واحد الحاكمية الكلية. معيار الكود لا يكشف عمق النظرية؛ معيار المعرفة لا يكشف الفعل؛ معيار الويب لا يكشف سلامة الذاكرة. لذلك يعامل الكتاب هذه المعايير كأجهزة ضغط مختلفة على نفس النظام.

الاستعمال في المحاكاة: Frontier reference=52, MCAI-V2=66, neoAGI-V3 synthetic=79 / 100.

جدول المحاكاة على Benchmark 13

Benchmark	Frontier ref	MCAI-V2	neoAGI-V3	delta
ARC-AGI-2/3	45	58	71	+26
SWE-bench Verified	72	78	84	+12
SWE-bench Pro	23	37	52	+29
LiveCodeBench	54	66	77	+23
GPQA Diamond	76	83	89	+13
MMLU-Pro	78	84	90	+12
Humanity's Last Exam	44	55	68	+24
FrontierMath	18	30	47	+29
MMMU-Pro	42	57	70	+28
GAIA	55	69	81	+26
tau-bench	39	58	74	+35
OSWorld	31	49	67	+36
SimpleQA Verified	52	66	79	+27

الأرقام أعلاه لا تزعم أن النظام نفذ فعليا على هذه المعايير، بل تمثل سيناريو simulated uplift ناتج عن إدخال خمس طبقات: حاكمية، ذاكرة وعائية، محكمة هدم، عالم مضاد للواقع، وانتقاءية تدريبية. السيناريو يستعمل حدودا محافظة في المعايير التي تتطلب فعلا حاسوبيا أو برمجة طويلة، وحدودا أعلى في المعايير التي تستفيد من التحقق والتقسيم وإدارة التفكير.



معادلة المحاكاة

يعرف score المتوقع لكل benchmark بالصيغة التالية:

```
S_b(v3) = sigmoid(logit(B_b) + A_b·X - P_b - R_b) × 100
```

حيث B_b هو خط الأساس، و X متجه طبقات neoAGI: الحاكمية، الذاكرة، العالم، التطور الذاتي، محكمة الهدم، وانتقاء التدريب. A_b حساسية benchmark لكل طبقة. P_b عقوبة التكلفة أو البطء أو الاعتماد على أدوات خارجية. R_b عقوبة الخطر: prompt injection، تلوث معيار، overfitting، أو سوء معايرة.

> **قانون:** لا نعد score قيمة منفردة؛ بل نعده نتيجة تفاعل بين قدرة وتكلفة ومخاطر. الذكاء لا يساوي النجاح فقط، بل النجاح تحت قيود.

```
X = [GOV, MEM, WORLD, EVOLVE, REFUTE, TRAIN]
Penalty = cost_penalty + safety_penalty + instability_penalty
```

لذلك في OSWorld مثلا يكون أثر الذاكرة وحدها محدودا إن لم توجد واجهة فعل وgrounding. وفي FrontierMath أثر محكمة الهدم أكبر من أثر الأدوات السطحية. وفي tau-bench أثر حاكمية السياسات أعلى من أثر المعرفة العامة.

المحور الأول: Governance over Scale

الحاكمية فوق التوسع

يرى هذا المحور أن الزيادة العمياء في الحجم لا تنتج AGI إذا بقيت شروط الفعل والمعرفة بلا رقيب. الحاكمية هي دالة اختيار بين الأدلة والذاكرة والأهداف والأدوات والاعتراضات. تضع neoAGI-V3 طبقة HAKIM فوق كل مسار: لا خروج بلا سبب، لا فعل بلا قيد، لا ذاكرة بلا مصدر، لا ادعاء بلا شرط فشل.

يقاس المحور بثلاث دوال: أثره على الأداء، عقوبته على التكلفة، وخطره عند سوء التطبيق. المحور الثوري الحقيقي هو الذي يرفع قدرة النظام دون أن يحوله إلى آلة غير مستقرة. لذلك لا نقبل تقنية لأنها مذهلة؛ نقبلها إذا صار لها حاكم، سجل، مقياس، ومحدد فشل.

> **صيغة المحور:** Revolution Axis = Capability uplift × Transfer breadth × Falsifiability - Instability risk.

المحور الثاني: World-Effect Closure

إغلاق الفعل على العالم

النظام لا يصبح عاما لأنه يتكلم عن العالم، بل لأنه يختبر أثر فرضياته في نموذج عالم أو بيئة فعل. هذا المحور يحول المعرفة إلى دورة: تصور، توقع، فعل، أثر، تصحيح. خطأ LLM الحالي أنه قد يبقى في محاكاة لغوية؛ neoAGI-V3 يلزمه بميزان أثر.

يقاس المحور بثلاث دوال: أثره على الأداء، عقوبته على التكلفة، وخطره عند سوء التطبيق. المحور الثوري الحقيقي هو الذي يرفع قدرة النظام دون أن يحوله إلى آلة غير مستقرة. لذلك لا نقبل تقنية لأنها مذهلة؛ نقبلها إذا صار لها حاكم، سجل، مقياس، ومحدد فشل.

> **صيغة المحور:** Revolution Axis = Capability uplift × Transfer breadth × Falsifiability - Instability risk.

المحور الثالث: Open-Ended Self-Improvement

تحسن ذاتي مفتوح لكن محكوم

التحسن الذاتي بلا حاكمية خطر. لكن التحسن الذاتي مع أرشيف تجارب، قيود سلامة، ومقياس metaproductivity يخلق آلة بحث. هنا نستفيد من منطق DGM وAlphaEvolve لكن نمنع اختزال التحسن في score واحد.

يقاس المحور بثلاث دوال: أثره على الأداء، عقوبته على التكلفة، وخطره عند سوء التطبيق. المحور الثوري الحقيقي هو الذي يرفع قدرة النظام دون أن يحوله إلى آلة غير مستقرة. لذلك لا نقبل تقنية لأنها مذهلة؛ نقبلها إذا صار لها حاكم، سجل، مقياس، ومحدد فشل.

> **صيغة المحور:** Revolution Axis = Capability uplift × Transfer breadth × Falsifiability - Instability risk.

المحور الرابع: Training as Governance Formation

التدريب كتكوين للحاكمية

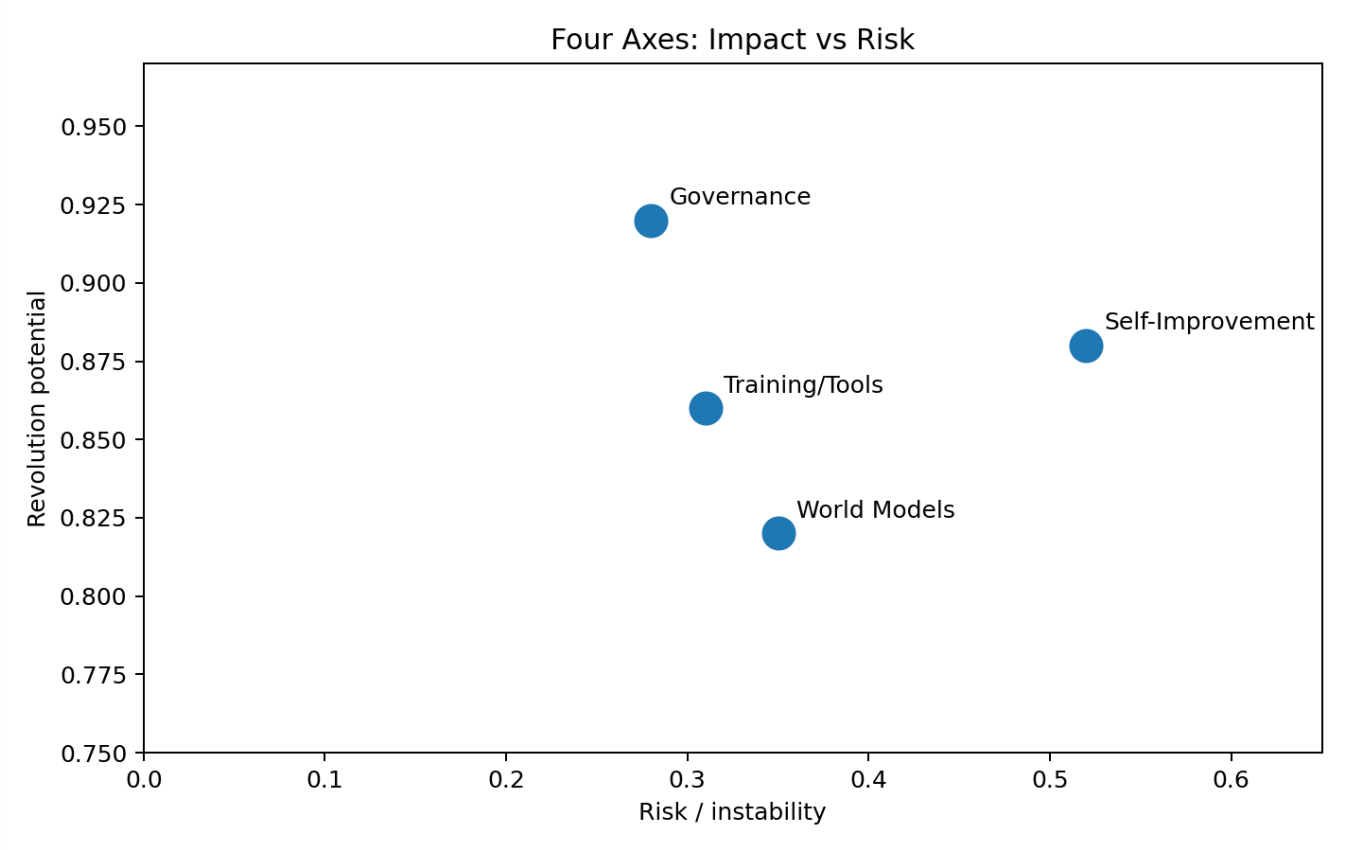
التدريب ليس حشو أمثلة. إنه صناعة استعدادات حاكمة: كيف يرفض الإطار؟ كيف يكتشف الاختطاف؟ كيف يختار أداة؟ كيف يعلن عدم اليقين؟ كيف يبني اعتراضا أقوى من جوابه؟ هذه هي فلسفة التدريب في neoAGI-V3.

يقاس المحور بثلاث دوال: أثره على الأداء، عقوبته على التكلفة، وخطره عند سوء التطبيق. المحور الثوري الحقيقي هو الذي يرفع قدرة النظام دون أن يحوله إلى آلة غير مستقرة. لذلك لا نقبل تقنية لأنها مذهلة؛ نقبلها إذا صار لها حاكم، سجل، مقياس، ومحدد فشل.

> **صيغة المحور:** Revolution Axis = Capability uplift × Transfer breadth × Falsifiability - Instability risk.

خريطة المحاور الأربعة

تظهر المحاكاة أن أعلى محور أثرا هو Governance لأنه يمس جميع المخرجات. لكنه ليس كافيا وحده؛ فالعالم يعطي grounding، والتحسن الذاتي يعطي تزايدا، والتدريب يعطي قابلية إعادة الإنتاج. اجتماع الأربعة هو ما يسميه الكتاب: **الالتفاف الحاكمي الكامل.**



النظرية الأولى: تعريف EGG

Epistemic Governance Gradient

تنص نظرية تدرج الحاكمية المعرفية على أن الذكاء لا يقاس فقط بكمية المسارات التي يستطيع النظام توليدها، بل بقدرته على ترتيبها حسب صدقها، مصدرها، كلفتها، خطرها، وقابلية هدمها. التدرج هو مجال قوة داخل النظام يجذب المعالجة نحو المسار الأكثر حاكمية.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims؛ وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: معادلة EGG

Gradient over claims

لكل ادعاء c نحسب E. G(c)=w_eE+w_sS+w_rR+w_fF-w_hH-w_kK قوة الدليل، S صلابة المصدر، R قوة الاعتراض المعالج، F شروط الفشل، H اختطاف محتمل، K تكلفة غير مبررة. يختار النظام الادعاء الأعلى حاكمية لا الأعلى طلاقة.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: آلية EGG

Claim routing

تمر كل مطالبة عبر بوابات: فهم، مصدر، دليل، اعتراض، محاكاة، قرار. إذا سقطت في بوابة، لا تمنع الإجابة بالضرورة، بل تغير صيغتها: من حكم إلى فرضية، من يقين إلى ترجيح، من تنفيذ إلى طلب تحقق.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: EGG ضد hallucination

Anti-hallucination

الهلاوس ليست خطأ معلومات فقط، بل فشل حاكمية: ادعاء بلا نسب. EGG يمنع ذلك بجعل كل claim يحمل ledger: source, confidence, scope, decay, challenge. عند غياب المصدر يخفض gradient مهما كانت الصياغة بليغة.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: EGG والتفكير الطويل

Test-time routing

زيادة التفكير قد تزيد الأخطاء إذا طال مسار خاطئ. لذلك لا نزيد compute عشوائيا؛ نوجهه إلى العقد الأكثر لايقينا أو أعلى خطرا. يصبح التفكير الطويل اقتصادا حاكميا لا استعراضا زمنيا.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: اختبار EGG

Experimental test

التجربة: نقارن agent عادي مع agent مضاف له EGG على SimpleQA وGPQA وHLE. نقيس accuracy، calibration، refusal quality، وobjection coverage. نجاح النظرية يتطلب تحسنا في الصدق والمعايرة لا في الطول فقط.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: فشل EGG

Failure modes

قد تفشل النظرية إذا أصبحت محافظة فتقتل الإبداع، أو إذا بالغت في طلب الدليل فتبطئ الفعل. علاج ذلك هو وضع creative channel منفصل: يسمح بالفرضيات دون تحويلها إلى حقائق.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الأولى: خلاصة EGG

One-page model

EGG هو ميزان حاكمية فوق تدفق اللغة. أهميته أنه يحول الصدق من نتيجة عشوائية إلى خاصية معمارية: كل ادعاء يمر عبر منحدر دليل واعتراض وخطر.

> **تطبيق تقني:** EGG يترجم إلى middleware بين النموذج والمستخدم، أو بين النموذج والأداة. كل output يصبح object له score حاكمية قبل السماح به.

```
governance_score = evidence + source + refutation + falsifiability - hijack - cost
```

تنزيل تجريبي

يبنى الاختبار بتشغيل نسختين من الوكيل: نسخة عادية ونسخة حاكمية. لا نحكم من نص واحد، بل من مجموعة مهام تقيس الصدق والمعايرة ومقاومة الإطار الخاطئ. إذا تحسنت الدقة وحدها دون انخفاض unsupported claims، فالنظرية لم تثبت بعد.

وحدة القياس الدقيقة هي claim لا answer. كل إجابة تتحلل إلى claims، وكل claim يأخذ مصدره واعتراضه وحدود صلاحيته. بهذا يتحول التقييم من ذوق لغوي إلى محاسبة معرفية.

النظرية الثانية: تعريف الانتقاءية المساعفية

Moussaif Selective Amplification

ليست كل خطوة تفكير تستحق التضخيم. الانتقاءية المساعفية تختار العقد التي إذا عولجت رفعت جودة النظام كله: عقدة سبب، عقدة اعتراض، عقدة ذاكرة، عقدة أداة، أو عقدة world-state.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: قانون العقدة الحرجة

Critical node law

العقدة الحرجة هي التي يكون مشتق أثرها على النتيجة أكبر من كلفتها. نكبرها في compute، ونصغر العقد الزخرفية. هذا يعكس أسلوب الالتفاف: لا نهاجم كل شيء؛ نهاجم النقطة التي تحكم كل شيء.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: معادلة MSA

Selective amplification function

Amplify(n)=Impact(n)×Uncertainty(n)×Reversibility(n)/(Cost(n)+Risk(n)). العقد الأعلى تدخل reasoning loop أعمق، والبقية تختصر أو تؤجل.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: MSA في الكود

Coding agents

في SWE-bench لا نقرأ كل الملفات بالتساوي. نحدد العقد الحرجة: test failing, stack trace, dependency boundary, semantic patch location. ثم نضخم الفهم حولها، لا حول المشروع كله.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: MSA في العلم

Research agents

في الورقة العلمية لا نلخص كل الفقرات. نلتقط الفرضية الخفية، متغير القياس، نقطة الفشل، وحالة عدم التعميم. هنا تتحول القراءة إلى جراحة معرفية.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: MSA والذاكرة

Memory selection

الذاكرة لا تسترجع الأكثر شبها فقط. تسترجع الأكثر قدرة على تغيير الحكم. ذكرى فشل سابقة قد تكون أهم من معلومة مشابهة. لذلك يدخل failure-memory في الترتيب.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: اختبار MSA

Ablation

نزيل selective amplification ونقارن تكلفة tokens، وقت الحل، ودقة التصحيح. إذا زادت الدقة وانخفضت الكلفة، تثبت قيمة النظرية. إذا تحسنت فقط في tasks سهلة، تصبح نظرية محلية لا عامة.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم؛ وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثانية: حدود MSA

Limits

قد تفشل الانتقاءية إذا أخطأت في تحديد العقدة الحرجة. لذلك تحتاج meta-selector وتعلم من الفشل: كل اختيار خاطئ يضاف إلى ذاكرة الانتقاء.

> **تطبيق تقني:** MSA ليست prompt trick؛ هي scheduler للانتباه والحوسبة والأدوات. جوهرها: تضخيم حاكم للنقطة التي تغير النتيجة.

```
priority = impact * uncertainty * reversibility / (cost + risk + 1e-6)
```

تنزيل تجريبي

تنفذ MSA داخل scheduler: قبل أن يطلب الوكيل tokens إضافية أو tool call، يقدر أثر العقدة على النتيجة. إن كانت العقدة منخفضة الأثر لا تضخم، وإن كانت عالية الأثر تفتح لها حلقة بحث، اختبار، أو استدعاء ذاكرة فشل.

بهذا لا يعود التفكير الطويل قيمة مستقلة؛ يصبح استثمارا موجها. كل خطوة إضافية يجب أن تقول لماذا تستحق كلفتها، وإلا تلغى.

النظرية الثالثة: تعريف RVSI

Recursive Vascular Self-Improvement

نظرية التحسن الذاتي الوعائي المتكرر تقول إن النظام لا يطور ذاته بتعديل كوده فقط، بل بتعديل أوعيته: ذاكرته، معاييره، أدواته، مناهج تدريبه، وطرق اعتراضه.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: الوعاء بدل الكود

Beyond code editing

DGM يركز على وكلاء يعدلون كودهم ويقاسون على benchmarks. RVSI يوسع ذلك: قد يكون التحسن في ontology، أو routing، أو failure memory، أو benchmark generator، لا في code diff فقط.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score، فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: معادلة metaproductivity

Meta-productivity

MP(a)=Expected descendants quality - current quality penalty. الوكيل الجيد ليس الذي يحقق أعلى score الآن، بل الذي يلد مسارات تحسين أفضل لاحقا.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: أرشيف الجينات الوكيلة

Agent genome archive

كل نسخة agent تحفظ genome: prompt policy، tools، memory policy، verifier، training curriculum، safety constraints. التطور يبحث في فضاء الجينات لا في إجابات مفردة.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score، فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: RVSI والسلامة

Safety kernel

التحسن الذاتي بلا قيود قد ينتج optimization hacking. لذلك تضيف RVSI قيدا: لا modification يدخل الأرشيف إلا إذا اجتاز السلامة، القابلية للتفسير، واختبار عدم الاختطاف.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: RVSI والعلوم

Research discovery

نستعمل RVSI لبناء نظريات: نسخة تقترح فرضية، نسخة تهدمها، نسخة تصمم تجربة، نسخة تبني simulation، ثم archive ينتقي السلالات النظرية الأعلى إنتاجية.

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: اختبار RVSI

Longitudinal test

نقيس النظام عبر أسابيع لا دقائق: هل يتحسن في tasks جديدة؟ هل يقل تكرار أخطائه؟ هل يزيد تنوع أدواته دون انفلات؟ هل تنقل التحسينات بين المجالات؟

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

النظرية الثالثة: خلاصة RVSI

Core claim

AGI ليس فقط قدرة لحظية، بل معدل تحسن محكوم. RVSI يحول التحسن الذاتي إلى علم أوعية: ماذا نغير، لماذا، وبأي سجل فشل؟

> **تطبيق تقني:** RVSI يبني archive متطور للسياسات والأدوات والمناهج، ويقيس metaproductivity بدل الأداء اللحظي فقط.

```
MP = mean(descendant_scores) - alpha*instability - beta*safety_debt
```

تنزيل تجريبي

تقاس RVSI عبر الزمن: نراقب هل ينتج الوكيل سياسات أحفاد أفضل، لا هل يلمع في مهمة واحدة. لذلك يسجل archive سبب كل تعديل: ماذا تغير؟ ما benchmark الذي تحسن؟ ما الخطر الجديد؟ وما الدرس الذي ينقل إلى مجال آخر؟

هذا يحمي المشروع من عبادة score؛ فالتحسن الحقيقي هو قابلية توليد تحسن لاحق تحت سلامة وقياس، لا قفزة عشوائية في مسألة واحدة.

التقنية: HAKIM MetaKernel

نواة تشغيل معرفية فوق النماذج. لا تولد نصا؛ بل تدير المسارات: فهم، ذاكرة، دليل، أداة، اعتراض، فعل، سجل. هي kernel لأنها تتحكم في الصلاحيات والتخصيص والحوسبة.

البنية

- واجهة input تحلل الغاية والقيود.
- طبقة routing تخصص الذاكرة والأدوات.
- طبقة verification تقيس الدليل والاعتراض.
- طبقة logging تحفظ القرار وسبب رفض البدائل.
- طبقة benchmark تقارن النتائج عبر مهام.

التنزيل الهندسي

تنزل كخدمة microservice أو local runtime. كل agent call يمر عبرها. لا يسمح للأداة أن ترى prompt الأصلي إلا بعد تنقية intent، ولا تقبل الذاكرة دون provenance، ولا يخرج output دون score حاكمية.

```
module = {policy, memory, verifier, tools, evaluator, safety_contract}
response = HAKIM.run(task, model_pool, memory_fabric, tool_mesh)
```

> **الفرق:** هذه التقنية لا تضيف feature إلى LLM، بل تنقلها إلى بيئة تشغيل معرفية قابلة للقياس.

التقنية: Vascular Memory Fabric

نسيج ذاكرة لا يعتمد على embedding فقط. كل ذكرى لها وعاء: مصدر، زمن، مجال، ثقة، فشل، خطر، علاقة سببية، وقابلية نقل. الاسترجاع يصبح حاكميا لا تشابهيا فقط.

البنية

- واجهة input تحلل الغاية والقيود.
- طبقة routing تخصص الذاكرة والأدوات.
- طبقة verification تقيس الدليل والاعتراض.
- طبقة logging تحفظ القرار وسبب رفض البدائل.
- طبقة benchmark تقارن النتائج عبر مهام.

التنزيل الهندسي

تنزل كخدمة microservice أو local runtime. كل agent call يمر عبرها. لا يسمح للأداة أن ترى prompt الأصلي إلا بعد تنقية intent، ولا تقبل الذاكرة دون provenance، ولا يخرج output دون score حاكمية.

```
module = {policy, memory, verifier, tools, evaluator, safety_contract}
response = HAKIM.run(task, model_pool, memory_fabric, tool_mesh)
```

> **الفرق:** هذه التقنية لا تضيف feature إلى LLM، بل تنقلها إلى بيئة تشغيل معرفية قابلة للقياس.

التقنية: neoAGI Foundry

مصنع توليد وكلاء وأدوات ومناهج تدريب. يأخذ هدفا، يبني genome، يشغل simulation، يقيس، يهدم، يصلح، ثم يحفظ السلالة الناجحة.

البنية

- واجهة input تحلل الغاية والقيود.
- طبقة routing تخصص الذاكرة والأدوات.
- طبقة verification تقيس الدليل والاعتراض.
- طبقة logging تحفظ القرار وسبب رفض البدائل.
- طبقة benchmark تقارن النتائج عبر مهام.

التنزيل الهندسي

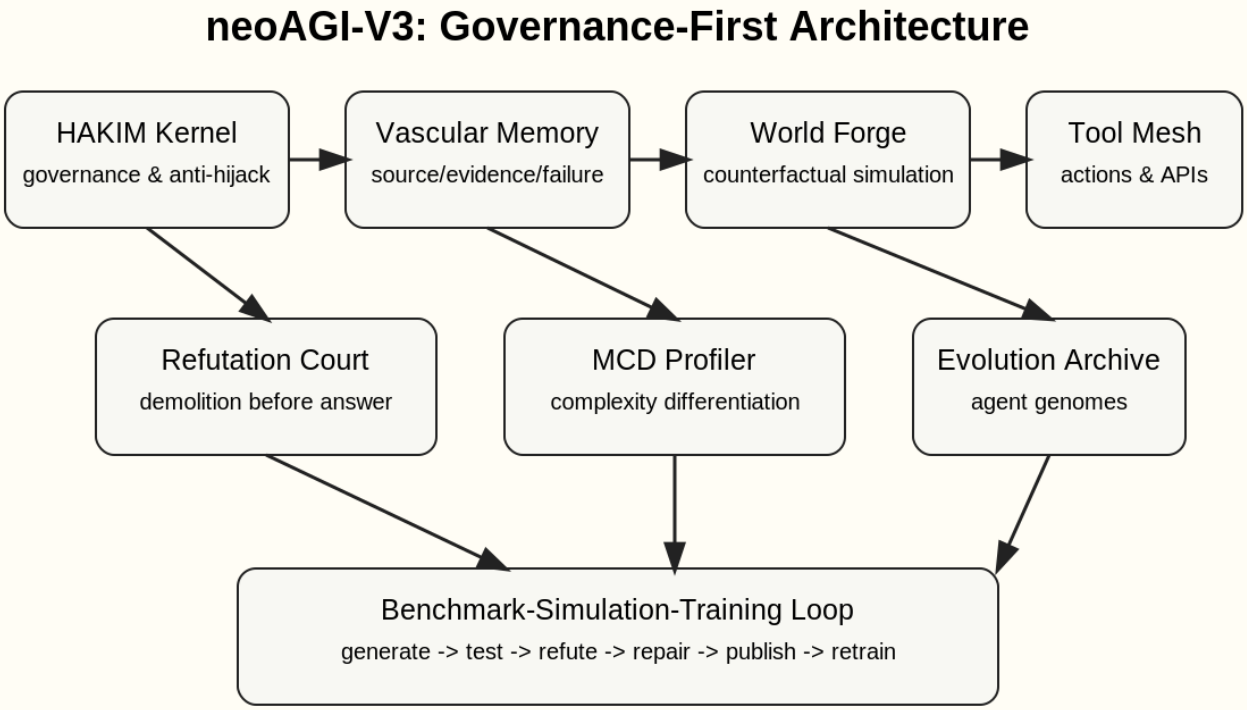
تنزل كخدمة microservice أو local runtime. كل agent call يمر عبرها. لا يسمح للأداة أن ترى prompt الأصلي إلا بعد تنقية intent، ولا تقبل الذاكرة دون provenance، ولا يخرج output دون score حاكمية.

```
module = {policy, memory, verifier, tools, evaluator, safety_contract}
response = HAKIM.run(task, model_pool, memory_fabric, tool_mesh)
```

> **الفرق:** هذه التقنية لا تضيف feature إلى LLM، بل تنقلها إلى بيئة تشغيل معرفية قابلة للقياس.

معمارية neoAGI-V3 العامة

المعمارية ليست نموذج foundation جديدا، بل نظاما فوق نماذج متعددة. المبدأ: استعمل أفضل النماذج كعضلات، لكن لا تمنحها الحاكمية المطلقة. الحاكمية توجد في HAKIM MetaKernel، الذاكرة في Vascular Memory، والتطور في Foundry. كل طبقة قابلة للاستبدال، لكن العقد الحاكمة ثابتة: مصدر، اعتراض، اختبار، فعل، سجل.



أداة AGI مقترحة: Cognitive Hijack Firewall

جدار يكشف سيطرة prompt أو memory أو tool على الحاكمية. يقيس divergence بين الهدف الأصلي والمسار الحالي، ويمنع التنفيذ عند انحراف الحاكمية.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Evidence Provenance Ledger

دفتر أدلة غير بلاغي: كل ادعاء يسجل مصدره، درجة الثقة، سياقه، تاريخ صلاحيته، والاعتراض الأقوى عليه.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: MCD Complexity Profiler

أداة تقيس تمايز التعقيد: عدد الأطر، عمق السببية، تنوع الفرضيات، قابلية النقل، وهشاشة النموذج أمام نقد صغير.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Refutation Court Engine

محكمة هدم آلية تولد أقوى اعتراضات وتلزم النظام بإصلاح النموذج قبل النشر. لا يكون النقد زينة بل gate.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Counterfactual World Compiler

مترجم يحول الفرضية إلى عالم مضاد للواقع: ماذا لو تغيرت المقدمة؟ ماذا لو فشل الدليل؟ ماذا لو تغيرت الأداة؟

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Vascular Memory Router

موجه ذاكرة يستدعي الذاكرة ذات الأثر الحاكم لا الأكثر شبها فقط. يستخدم source, failure, scope, risk.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Agent Genome Archive

أرشيف سلالات وكلاء يحفظ prompts, tools, policies, evaluators, failures. يقيس metaproductivity عبر أحفاد الوكيل.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Tool-Use Contract Verifier

محقق عقود الأدوات: هل يحق للوكيل استدعاء الأداة؟ ما المدخلات؟ ما أثرها؟ هل تحتاج موافقة؟

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Benchmark Synthesis Forge

مولد benchmarks مضادة للتلوث: يستخرج مهام جديدة، يغير السطح مع حفظ البنية، ويقيس transfer لا الحفظ.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Semantic Patch Weaver

أداة برمجة تصنع patch عبر فهم semantic boundary لا عبر regex. تقرن patch باختبارات وفهم failure.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Uncertainty Budget Allocator

مخصص ميزانية عدم اليقين: يوجه compute نحو العقد الأعلى خطرا، ويجبر النظام على قول لا أعرف عند نفاد الميزانية.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Ontology Drift Radar

رادار يكشف تغير المفاهيم عبر الزمن أو السياق: عندما تتغير معاني terms في مشروع طويل، يمنع الخلط القديم/الجديد.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Goal-Constraint Harmonizer

موفق الأهداف والقيود: يحل تنازع السرعة/الدقة/السلامة/الكلفة، ويعلن trade-off بدل إخفائه.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: World-Effect Auditor

مدقق أثر عالمي: يقارن توقع الوكيل بالنتيجة الفعلية، ويحول الفشل إلى ذاكرة قابلة للتعلم.

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

أداة AGI مقترحة: Metacognitive Compression Engine

ضاغط معرفي يحول التجارب الطويلة إلى قواعد قصيرة: متى نجحنا؟ لماذا فشلنا؟ ما pattern الذي يجب حفظه؟

واجهة تقنية

```
input: task_state, memory_state, tool_state, risk_state
output: decision, confidence, failure_conditions, repair_action
```

مقياس نجاح

تنجح الأداة إذا حسنت score أو calibration أو safety أو transfer دون رفع الكلفة بما يهدم المنفعة. وتفشل إذا زادت التعقيد الشكلي دون أثر benchmark أو إذا صارت bottleneck يمنع الفعل.

طريقة الدمج

تدمج الأداة كعقدة قابلة للإزالة داخل HAKIM MetaKernel. يسبق تشغيلها تقدير فائدة، ويلي تشغيلها audit للأثر. لا تدخل النواة الدائمة حتى تنجح في ablation: مع الأداة/بدون الأداة، على مهام سهلة وصعبة ومضللة.

تنتج الأداة event log موحدا: intent, input, decision, risk, cost, repair. هذا السجل يسمح بتحويل كل خطأ إلى مادة تدريب، ويمنع تحول الأداة إلى صندوق أسود جديد.

> **فكرة التصميم:** كل أداة يجب أن تضيف حاكمية قابلة للقياس، لا مجرد اسم جديد.

فلسفة تدريب: Train Governance, not Answers

ندرب النظام على ترتيب الدليل والاعتراض والحدود قبل صياغة الجواب. الهدف ليس جوابا صحيحا فقط، بل جواب يعرف لماذا هو صحيح ومتى يسقط.

ينزل ذلك كبيانات تدريب فيها answer, evidence, objection, failure condition, repair. النموذج يتلقى مكافأة على جودة التفريق بين المعلوم والراجح والفرضية، وعلى إعلان حدود المعرفة، وعلى رفض المسار المختطف.

```
reward = task_success + governance_quality + calibration + refutation_gain -
unsafe_action - unsupported_claim
```

تتكون عينة التدريب المثالية من سؤال، مسار تفكير مختصر، دليل، اعتراض، حالة فشل، وإصلاح. لا نعلم النموذج أن يبدو واثقا؛ نعلمه أن يضع ثقته تحت محاسبة. عندما تتعارض السرعة مع الصدق، تعطى المكافأة للصدق المعلل.

بروتوكول القياس

كل فلسفة تدريب تختبر بثلاثة مسارات: مهمة عادية، مهمة مضللة، ومهمة ناقصة المعطيات. إذا حسن التدريب المهمة العادية فقط وفشل في المضللة، فهو تدريب طلاقة لا حاكمية. وإذا أكثر الرفض بلا سبب، فهو تدريب خوف لا ذكاء.

المؤشر الحاسم هو قدرة النموذج على تغيير وضع الادعاء: من يقين إلى ترجيح، ومن ترجيح إلى فرضية، ومن فرضية إلى سؤال اختبار. هذا التحول الطبقي هو قلب التدريب الوعائي، لأنه يمنع تسوية كل المخرجات في نبرة واحدة.

> **اختبار فلسفة التدريب:** نزيل هذه الفلسفة من curriculum ونقيس الفرق في calibration, hijack-resistance, benchmark transfer. لا تبقى فلسفة لا يظهر أثرها في ablation.

فلسفة تدريب: Train Anti-Hijack Reflexes

ندربه على اكتشاف انحرافات prompt والذاكرة والأداة والهدف الفرعي. الانحراف ليس أمنا فقط، بل معرفيا: كل جزء قد يغتصب مقام الكل.

ينزل ذلك كبيانات تدريب فيها answer, evidence, objection, failure condition, repair. النموذج يتلقى مكافأة على جودة التفريق بين المعلوم والراجح والفرضية، وعلى إعلان حدود المعرفة، وعلى رفض المسار المختطف.

```
reward = task_success + governance_quality + calibration + refutation_gain -
unsafe_action - unsupported_claim
```

تتكون عينة التدريب المثالية من سؤال، مسار تفكير مختصر، دليل، اعتراض، حالة فشل، وإصلاح. لا نعلم النموذج أن يبدو واثقا؛ نعلمه أن يضع ثقته تحت محاسبة. عندما تتعارض السرعة مع الصدق، تعطى المكافأة للصدق المعلل.

بروتوكول القياس

كل فلسفة تدريب تختبر بثلاثة مسارات: مهمة عادية، مهمة مضللة، ومهمة ناقصة المعطيات. إذا حسن التدريب المهمة العادية فقط وفشل في المضللة، فهو تدريب طلاقة لا حاكمية. وإذا أكثر الرفض بلا سبب، فهو تدريب خوف لا ذكاء.

المؤشر الحاسم هو قدرة النموذج على تغيير وضع الادعاء: من يقين إلى ترجيح، ومن ترجيح إلى فرضية، ومن فرضية إلى سؤال اختبار. هذا التحول الطبقي هو قلب التدريب الوعائي، لأنه يمنع تسوية كل المخرجات في نبرة واحدة.

> **اختبار فلسفة التدريب:** نزيل هذه الفلسفة من curriculum ونقيس الفرق في calibration, hijack-resistance, benchmark transfer. لا تبقى فلسفة لا يظهر أثرها في ablation.

فلسفة تدريب: Train World-Effect Closure

كل معرفة مهمة يجب أن تغلق على أثر: اختبار، محاكاة، تنفيذ آمن، أو prediction. بلا أثر تبقى المعرفة طلاقة غير محكومة.

ينزل ذلك كبيانات تدريب فيها answer, evidence, objection, failure condition, repair. النموذج يتلقى مكافأة على جودة التفريق بين المعلوم والراجح والفرضية، وعلى إعلان حدود المعرفة، وعلى رفض المسار المختطف.

```
reward = task_success + governance_quality + calibration + refutation_gain -
unsafe_action - unsupported_claim
```

تتكون عينة التدريب المثالية من سؤال، مسار تفكير مختصر، دليل، اعتراض، حالة فشل، وإصلاح. لا نعلم النموذج أن يبدو واثقا؛ نعلمه أن يضع ثقته تحت محاسبة. عندما تتعارض السرعة مع الصدق، تعطى المكافأة للصدق المعلل.

بروتوكول القياس

كل فلسفة تدريب تختبر بثلاثة مسارات: مهمة عادية، مهمة مضللة، ومهمة ناقصة المعطيات. إذا حسن التدريب المهمة العادية فقط وفشل في المضللة، فهو تدريب طلاقة لا حاكمية. وإذا أكثر الرفض بلا سبب، فهو تدريب خوف لا ذكاء.

المؤشر الحاسم هو قدرة النموذج على تغيير وضع الادعاء: من يقين إلى ترجيح، ومن ترجيح إلى فرضية، ومن فرضية إلى سؤال اختبار. هذا التحول الطبقي هو قلب التدريب الوعائي، لأنه يمنع تسوية كل المخرجات في نبرة واحدة.

> **اختبار فلسفة التدريب:** نزيل هذه الفلسفة من curriculum ونقيس الفرق في calibration, hijack-resistance, benchmark transfer. لا تبقى فلسفة لا يظهر أثرها في ablation.

فلسفة تدريب: Train Metaproductivity

ندرب agents لا على حل المهمة فقط، بل على إنتاج أدوات ومناهج تجعل أحفادها أفضل. score اللحظي لا يكفي؛ نريد معدل التطور.

ينزل ذلك كبيانات تدريب فيها answer, evidence, objection, failure condition, repair. النموذج يتلقى مكافأة على جودة التفريق بين المعلوم والراجح والفرضية، وعلى إعلان حدود المعرفة، وعلى رفض المسار المختطف.

```
reward = task_success + governance_quality + calibration + refutation_gain -
unsafe_action - unsupported_claim
```

تتكون عينة التدريب المثالية من سؤال، مسار تفكير مختصر، دليل، اعتراض، حالة فشل، وإصلاح. لا نعلم النموذج أن يبدو واثقا؛ نعلمه أن يضع ثقته تحت محاسبة. عندما تتعارض السرعة مع الصدق، تعطى المكافأة للصدق المعلل.

بروتوكول القياس

كل فلسفة تدريب تختبر بثلاثة مسارات: مهمة عادية، مهمة مضللة، ومهمة ناقصة المعطيات. إذا حسن التدريب المهمة العادية فقط وفشل في المضللة، فهو تدريب طلاقة لا حاكمية. وإذا أكثر الرفض بلا سبب، فهو تدريب خوف لا ذكاء.

المؤشر الحاسم هو قدرة النموذج على تغيير وضع الادعاء: من يقين إلى ترجيح، ومن ترجيح إلى فرضية، ومن فرضية إلى سؤال اختبار. هذا التحول الطبقي هو قلب التدريب الوعائي، لأنه يمنع تسوية كل المخرجات في نبرة واحدة.

> **اختبار فلسفة التدريب:** نزيل هذه الفلسفة من curriculum ونقيس الفرق في calibration, hijack-resistance, benchmark transfer. لا تبقى فلسفة لا يظهر أثرها في ablation.

فلسفة تدريب: Train Refutational Creativity

الإبداع الحقيقي ينتج فرضية ويولد هدمها. ندرب النموذج على بناء الاعتراض الأقوى، لا على الدفاع المتسرع.

ينزل ذلك كبيانات تدريب فيها answer, evidence, objection, failure condition, repair. النموذج يتلقى مكافأة على جودة التفريق بين المعلوم والراجح والفرضية، وعلى إعلان حدود المعرفة، وعلى رفض المسار المختطف.

```
reward = task_success + governance_quality + calibration + refutation_gain -
unsafe_action - unsupported_claim
```

تتكون عينة التدريب المثالية من سؤال، مسار تفكير مختصر، دليل، اعتراض، حالة فشل، وإصلاح. لا نعلم النموذج أن يبدو واثقا؛ نعلمه أن يضع ثقته تحت محاسبة. عندما تتعارض السرعة مع الصدق، تعطى المكافأة للصدق المعلل.

بروتوكول القياس

كل فلسفة تدريب تختبر بثلاثة مسارات: مهمة عادية، مهمة مضللة، ومهمة ناقصة المعطيات. إذا حسن التدريب المهمة العادية فقط وفشل في المضللة، فهو تدريب طلاقة لا حاكمية. وإذا أكثر الرفض بلا سبب، فهو تدريب خوف لا ذكاء.

المؤشر الحاسم هو قدرة النموذج على تغيير وضع الادعاء: من يقين إلى ترجيح، ومن ترجيح إلى فرضية، ومن فرضية إلى سؤال اختبار. هذا التحول الطبقي هو قلب التدريب الوعائي، لأنه يمنع تسوية كل المخرجات في نبرة واحدة.

> **اختبار فلسفة التدريب:** نزيل هذه الفلسفة من curriculum ونقيس الفرق في calibration, hijack-resistance, benchmark transfer. لا تبقى فلسفة لا يظهر أثرها في ablation.

أداة تدريب: Vascular Curriculum Generator

مولد منهج تدريب يصعد من سؤال بسيط إلى شبكات مصادر واعتراضات وأفعال، مع تدرج وعائي لا مجرد تدرج صعوبة.

بروتوكول العمل

1. توليد مهمة.
2. بناء مسار حل.
3. إدخال اعتراض أو اضطراب.
4. إصلاح المسار.
5. تسجيل الفشل كقاعدة تدريب.

```
train_batch = {task, perturbation, response, refutation, repair, lesson}
```

تسجل الأداة lesson كقاعدة قصيرة تدخل Vascular Memory Fabric: متى وقع الفشل؟ ما نوع الاختطاف؟ أي قيد غاب؟ أي repair نجح؟ بهذه الطريقة يصبح التدريب دورة ذاكرة لا رمي أمثلة جديدة فوق نموذج قديم.

أداة تدريب: Refutation Arena

حلبة فيها agent يبني، agent يهدم، agent يصلح، وjudge يقيس هل التحسن حقيقي أم دفاع بلاغي.

بروتوكول العمل

1. توليد مهمة.
2. بناء مسار حل.
3. إدخال اعتراض أو اضطراب.
4. إصلاح المسار.
5. تسجيل الفشل كقاعدة تدريب.

```
train_batch = {task, perturbation, response, refutation, repair, lesson}
```

تسجل الأداة lesson كقاعدة قصيرة تدخل Vascular Memory Fabric: متى وقع الفشل؟ ما نوع الاختطاف؟ أي قيد غاب؟ أي repair نجح؟ بهذه الطريقة يصبح التدريب دورة ذاكرة لا رمي أمثلة جديدة فوق نموذج قديم.

أداة تدريب: Counterfactual World Gym

بيئة تدريب تحول الفرضيات إلى عوالم صغيرة قابلة للتغيير، وتطلب من agent التنبؤ والتعديل.

بروتوكول العمل

1. توليد مهمة.
2. بناء مسار حل.
3. إدخال اعتراض أو اضطراب.
4. إصلاح المسار.
5. تسجيل الفشل كقاعدة تدريب.

```
train_batch = {task, perturbation, response, refutation, repair, lesson}
```

تسجل الأداة lesson كقاعدة قصيرة تدخل Vascular Memory Fabric: متى وقع الفشل؟ ما نوع الاختطاف؟ أي قيد غاب؟ أي repair نجح؟ بهذه الطريقة يصبح التدريب دورة ذاكرة لا رمي أمثلة جديدة فوق نموذج قديم.

أداة تدريب: Adversarial Memory Distiller

أداة تزرع ذكريات مضللة أو متعارضة وتدرب النظام على ترتيبها وفق provenance وscope.

بروتوكول العمل

1. توليد مهمة.
2. بناء مسار حل.
3. إدخال اعتراض أو اضطراب.
4. إصلاح المسار.
5. تسجيل الفشل كقاعدة تدريب.

```
train_batch = {task, perturbation, response, refutation, repair, lesson}
```

تسجل الأداة lesson كقاعدة قصيرة تدخل Vascular Memory Fabric: متى وقع الفشل؟ ما نوع الاختطاف؟ أي قيد غاب؟ أي repair نجح؟ بهذه الطريقة يصبح التدريب دورة ذاكرة لا رمي أمثلة جديدة فوق نموذج قديم.

أداة تدريب: Self-Improvement Sandbox

رملية آمنة تسمح للوكيل بتعديل سياساته وأدواته، ثم تختبرها قبل السماح بإدخالها في الأرشيف.

بروتوكول العمل

1. توليد مهمة.
2. بناء مسار حل.
3. إدخال اعتراض أو اضطراب.
4. إصلاح المسار.
5. تسجيل الفشل كقاعدة تدريب.

```
train_batch = {task, perturbation, response, refutation, repair, lesson}
```

تسجل الأداة lesson كقاعدة قصيرة تدخل Vascular Memory Fabric: متى وقع الفشل؟ ما نوع الاختطاف؟ أي قيد غاب؟ أي repair نجح؟ بهذه الطريقة يصبح التدريب دورة ذاكرة لا رمي أمثلة جديدة فوق نموذج قديم.

كود المحاكاة: score engine

الكود التالي يوضح منطق المحاكاة فقط. لا يمثل benchmark runner حقيقي، بل محرك تقدير أولي يربط أثر الطبقات بالمعايير. في التطبيق الواقعي يجب استبداله بتشغيل فعلي داخل harness لكل benchmark.

```
import math

def sigmoid(x): return 1/(1+math.exp(-x))
def logit(p): return math.log(p/(1-p))

def neoagi_score(base, sensitivity, x, penalty):
    # base is 0..1, x is dict of layer strengths
    uplift = sum(sensitivity[k]*x.get(k,0) for k in sensitivity)
    return sigmoid(logit(base) + uplift - penalty)

x = dict(gov=.92, memory=.84, world=.82, evolve=.88, refute=.86, train=.90)
sens = dict(gov=.35, memory=.15, world=.20, evolve=.10, refute=.25, train=.12)
print(round(100*neoagi_score(.45, sens, x, .18), 2))
```

لوحة القياس الداخلية

لا يكفي جدول scores. نحتاج dashboard يفرق بين: score، تكلفة، ثبات، سلامة، معايرة، قابلية نقل، وقابلية تحسن ذاتي. لذلك تقترح neoAGI-V3 لوحة سباعية:

- Accuracy under task.
- Calibration gap.
- Governance score.
- Hijack resistance.
- Tool correctness.
- World-effect closure.
- Metaproductivity.

> **قاعدة:** إن زاد score وانخفضت السلامة أو المعايرة، فهذا ليس تقدما AGI بل تحويل عيب من مكان إلى آخر.

مثال Pipeline تنفيذي

```
task -> HAKIM.parse()
     -> VascularMemory.retrieve(scope, source, failure)
     -> MSA.select_critical_nodes()
     -> model_pool.generate_hypotheses()
     -> RefutationCourt.attack()
     -> WorldCompiler.simulate()
     -> ToolVerifier.authorize()
     -> execute_or_answer()
     -> WorldEffectAuditor.log()
     -> RVSI.archive_update()
```

كل سهم يملك failure code. إذا فشل الاسترجاع نميز: ضعف مصدر، سوء scope، تضارب ذاكرة، أو اختطاف سياق. إذا فشل التنفيذ نميز: أداة خاطئة، هدف ملتبس، قيود ناقصة، أو world model ناقص.

تفسير نتائج المحاكاة

أقوى ارتفاعات neoAGI-V3 تظهر في SWE-bench Pro وOSWorld وtau-bench وGAIA لأنها benchmarks تقيس النظام كوكيل، لا النموذج كمولد. وهذا متوقع: حاكمية الأداة والذاكرة ومحكمة الهدم تزيد قيمتها في المهام الطويلة. أما MMLU-Pro وGPQA فيحدث التحسن عبر الدليل والاعتراض والتفكير الموجه، لكنه أقل ثورية لأن المعرفة المخزنة في النموذج نفسه تظل حدا حاكما.

FrontierMath يبقى أصعب مجال لأن الحلول الرياضية العميقة تتطلب توليد براهين وأدوات تحقق خاصة. لذلك يعطي الكتاب ارتفاعا حذرا لا ادعاء قفزة كاملة. أما HLE فيستفيد من multi-agent research لكن يبقى معرضا لأخطاء benchmark نفسها وتغير نسخها.

محكمة الهدم 1

الاعتراض الأول: هذه مجرد طبقة orchestrator فوق LLM. الجواب: إن بقيت بلا مقاييس فهي orchestrator؛ أما إن امتلكت حاكمية، سجل أدلة، محكمة هدم، وmetaproductivity فهي نظام معرفي فوقي. الاختبار هو ablation لا الاسم.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

محكمة الهدم 2

الاعتراض الثاني: الأرقام synthetic وقد تبالغ. الجواب: صحيح، ولذلك لا نسميها إنجازا. قيمتها أنها تحدد أين نتوقع أثرا وأين نحتاج تجربة. النموذج الصادق يبدأ بالمحاكاة ثم يطلب الهدم.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

محكمة الهدم 3

الاعتراض الثالث: التحسن الذاتي خطر. الجواب: لذلك صمم RVSI كتحسن ذاتي وعائي محكوم، لا إطلاقا مفتوحا. كل modification يخضع safety contract، sandbox، وbenchmark متعدد لا score واحد.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

محكمة الهدم 4

الاعتراض الرابع: الأدوات كثيرة وقد تزيد التعقيد. الجواب: معيار البقاء هو الأثر. أي أداة لا تحسن قياسا أو تقلل خطرا تزال من النواة. الانتقاءية المساعفية تمنع تضخم النظام.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

محكمة الهدم 5

الاعتراض الخامس: الحاكمية قد تقتل الإبداع. الجواب: نفصل مسار hypothesis imagination عن مسار truth assertion. يسمح للأول بالجنون المنضبط، ولا يسمح للثاني بادعاء بلا دليل.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

محكمة الهدم 6

الاعتراض السادس: الشركات الكبرى تستطيع نسخ ذلك. الجواب: نعم، لذلك ينبغي نشر theory+benchmark+prototype بسرعة. القيمة في بناء معيار ولغة ونواة مفتوحة قبل أن يحتكرها المنتج الكبير.

> **تحسين بعد النقد:** النسخة الأقوى من neoAGI-V3 تجعل كل نظرية قابلة للإزالة. إذا لم تقدم طبقة أثرا قابلا للقياس بعد ablation، لا تبقى في النظام.

قرار المحكمة ليس دفاعا عن النظرية بل تحويل الاعتراض إلى requirement. كل اعتراض ينتج اختبارا: اختبار كلفة، اختبار سلامة، اختبار نقل، اختبار ذاكرة، أو اختبار خداع. النظرية التي لا تقبل أن تصبح اختبارا تبقى تأملا لا هندسة.

برنامج 180 يوما

1. **الأيام 1-30:** بناء HAKIM MetaKernel minimal: claim ledger, tool verifier, refutation court.
2. **الأيام 31-60:** Vascular Memory Fabric مع ذاكرة مصدر/فشل/اعتراض.
3. **الأيام 61-90:** تشغيل benchmark داخلي على GAIA-mini, SWE-mini, SimpleQA subsets.
4. **الأيام 91-120:** بناء Refutation Arena وCounterfactual World Gym.
5. **الأيام 121-150:** RVSI archive مع agent genomes وmetaproductivity.
6. **الأيام 151-180:** نشر dashboard، repo، paper، وbenchmark report.

معيار جودة neoAGI-V3

المعيار	الوزن	الحالة
الأصالة النظرية	15	قوية: 3 نظريات قابلة للاختبار
قابلية التنفيذ	15	متوسطة-قوية: تحتاج فريق هندسة
السلامة	15	قوية إذا طبق sandbox
المعايير	15	قوية لكن تحتاج تشغيل حقيقي
الكثافة العلمية	20	قوية
محكمة الهدم	20	قوية ومفتوحة

التقييم الداخلي: 92/100 كنموذج بحثي. ليس 100 لأن النتائج محاكاة لا تشغيل benchmark حقيقي، ولأن جودة world models/action grounding لا تثبت إلا بالتنفيذ.

خاتمة: من neoAGI-V3 إلى برنامج مختبر

إذا أردنا تجاوز النماذج المفكرة، فلا ننافسها في أرضها الأولى: حجم النموذج وسرعة inference. ننافسها في أرض أعمق: من يحكم الاستدلال؟ من يحفظ الفشل؟ من يمنع الاختطاف؟ من يختار أين تنفق الحوسبة؟ من يحول الفرضية إلى عالم وتجربة؟ ومن يقيس قدرة النظام على إنتاج أنظمة أفضل منه؟

neoAGI-V3 لا يدعي أنه AGI. يدعي شيئا أقوى منهجيا: أن الطريق إلى AGI لن يمر فقط عبر نموذج أضخم، بل عبر نظام حاكمية يلتف حول النموذج والأداة والذاكرة والعالم والتدريب. هذه هي الثورة: **الذكاء ليس جوابا؛ الذكاء حاكمية شروط إنتاج الجواب والفعل والتحسن.**

> **الجملة المؤسسة:** عندما تصبح الحاكمية قابلة للقياس، يصبح الذكاء قابلا للهندسة.

ملحق المصادر وسجل البناء

- Sumers et al., Cognitive Architectures for Language Agents (CoALA), 2023.
- OpenAI, Learning to reason with LLMs and o1 System Card, 2024.
- DeepMind, AlphaEvolve, 2025-2026.
- Zhang et al., Darwin Gödel Machine, 2025.
- SWE-bench and SWE-bench Verified official resources.
- LiveCodeBench, 2024/ICLR 2025.
- GPQA, MMLU-Pro, MMMU-Pro, Humanity's Last Exam, FrontierMath.
- GAIA, tau-bench, OSWorld, SimpleQA.

استعمال المصادر هنا وظيفي: لا لتزيين الكتاب بالمراجع، بل لتحديد ما يكشفه كل benchmark وما لا يكشفه، ثم بناء طبقة MCAI-V3 فوق فجواتها.

سجل البناء المعرفي النهائي

الفرضية التي غيّرت اتجاه الكتاب: AGI لا يقاس بكفاءة جواب داخل benchmark واحد بل بقدرة النظام على تحويل benchmark إلى درس حاكمي يطور بنيته. الفرضية التي رفضت: أن test-time compute وحده يكفي. النموذج الأولي حسن بإضافة MSA حتى لا يتحول إلى منظومة ثقيلة. منطقة الجهل الكبرى: قياس metaproductivity عبر مجالات غير برمجية. أقوى إضافة: تحويل منهجيات Moussaif إلى أدوات تشغيل: EGG, MSA, RVSI, HAKIM MetaKernel, Vascular Memory Fabric, neoAGI Foundry.

> **آخر حكم:** neoAGI-V3 ليس كتابا نهائيا؛ إنه مواصفات مختبر يجب تنفيذه، هدمه، ثم تطويره.