AGI-OS القابل للتكاثف وإعادة البناء

خمس نظريات مساعفية لتوحيد HyperKnowledge وHyperSkill وHyperAbility وHyperLoop والبروتوكولات فوق المعرفية

زهير مساعف — Zouhair Moussaif

13 أغسطس 2026

الإطار: MOUSSAIF AGI-Oriented Research Program / M-HKOS / HAKIM-MCD / MICAGFI lineage.

عقد الصدق العلمي

هذه الورقة تقترح بنية بحثية وهندسية ولا تدّعي أن AGI قد تحقق، ولا تنسب وعياً أو استقلالاً ذاتياً إلى نموذج أو منظومة. النظريات الخمس الواردة هنا **مقترحات أصلية ضمن هذا العمل** بمعنى أنها صيغت هنا كآليات مترابطة ذات معادلات أولية وأدوات وبروتوكولات واختبارات تفنيد، ولا يُدّعى لها سبق تاريخي مطلق من دون مسح منهجي شامل للأدبيات وبراءات الاختراع.

المعيار الحاكم هو: لا تُرقّى تسمية جديدة إلى «نظرية» ما لم تُنتج فرقاً آلياً قابلاً للاختبار، وتنبؤاً يختلف عن baseline واضح، وتجربة قد ترفضها. ولا تُرقّى قدرة إلى OS ما لم يثبت أثرها عبر النقل، والتحقق، وحدود الصلاحية، والاسترجاع، والرجوع.

الكلمات المفتاحية: MCD، HyperLoop، HyperAbility، HyperSkill، HyperKnowledge، KB-as-OS، AGI-oriented systems، Self-Reconstruction، World-Effect، Falsification Court، Computational Vascularity، HAKIM-MCD، Q-MCD، Memory.

الملخص والمساهمة

تتناول هذه الورقة فجوة تظهر عندما تُبنى الأنظمة الوكيلية بوصفها تجميعاً لمزايا منفصلة: نموذج لغوي، ذاكرة، أدوات، مخطط، عدة وكلاء، ثم طبقة تقييم. هذا التجميع قد يرفع القدرة، لكنه لا يجيب وحده عن سؤال أشد عمقاً: **كيف تتحول الفروق المعرفية التي تُحل مراراً إلى قدرة مستقرة، وكيف تعبر المعرفة بين الطبقات دون تلوث، وكيف نعرف أن مهارة ما تنتمي فعلاً إلى حلقة تشغيلية، وكيف نخصص الحساب وفق أثر سببي، وكيف تعيد المنظومة بناء ذاتها دون إعادة كتابة تاريخها؟**

نقترح خمس نظريات مترابطة. الأولى، **تكاثف الفروق الوكيلية ADC**، تعرّف القدرة بوصفها تكاثفاً لتسويات فروق فعالة ثبت أثرها ونقلها. الثانية، **النفاذية المعرفية المحكومة GEP**، تعامل انتقال المعرفة بين الذاكرة والمهارة والقرار كعملية عبور لها معامل نفاذية يعتمد على الدليل والمصدر والرجوع والتماسك. الثالثة، **رنين المهارة-الحلقة SLR**، تفترض أن HyperSkill لا تكون مستقرة إلا عندما يوجد رنين قابل للقياس بينها وبين HyperLoop يغلق أثراً خارجياً. الرابعة، **الجهد السببي الحاسوبي CCP**، تقترح توجيه compute نحو المسار ذي أعلى تغير عالمي هامشي متوقع بدلاً من توجيهه إلى ما يبدو لغوياً «أصعب». الخامسة، **إعادة البناء الذاتي المضاد للتاريخ الزائف CSR**، تجعل تحديث الذاكرة عملية تقدير حالة تحفظ السجل والتعارض وتنجح في إعادة لعب مضادة للواقع.

تُدمج النظريات داخل خط تشغيل واحد: السؤال الحاكم ← الفرق الفعال ← MCD/Q-MCD ← التروية الحاسوبية ← عقد المهارة/القدرة ← HyperLoop محدود ← محاكم التفنيد والهدم ← Memory Delta ← World-Effect Ledger ← إعادة البناء. وتقترح الورقة خمس أدوات وخمسة بروتوكولات، ومجموعة تجارب ablation وholdout ونقل يمكن أن ترفض كل نظرية بدلاً من جعلها محصنة لغوياً.

الفرضية الجامعة

إذا كانت النظريات الخمس صحيحة جزئياً على الأقل، فينبغي أن يظهر أثرها بصورة مركبة: تقل كلفة الاسترجاع والبحث حين تتكاثف الخبرة إلى skills، تنخفض أخطاء الانتشار حين تُحكم النفاذية، يقل الدوران حين يُقاس الرنين، يتحسن العائد لكل وحدة compute حين يوجّه الجهد السببي، ويصبح السجل أكثر قابلية لإعادة الإنتاج حين تمر الذاكرة بمحكمة مضادة للتاريخ الزائف. هذه فرضية مركبة يمكن تفكيكها بتجارب ablation مستقلة، ولا يشترط قبولها كوحدة واحدة.

1. المشكلة البحثية: من تجميع القدرات إلى نظام حاكم للقدرة

انتقل البحث في الوكلاء اللغويين من مجرد إنتاج النص إلى التخطيط، استخدام الأدوات، العمل في بيئات تفاعلية، حفظ الذاكرة، واختيار مسارات متعددة. ومع ذلك يمكن لنظام غني بالأدوات أن يظل هشاً إذا كانت طبقات المعرفة والمهارة والحلقات والذاكرة تعمل من دون علاقات ترخيص وأسبقية. الخطأ المركزي ليس دائماً نقص «الذكاء الخام»؛ قد يكون أن معرفة منخفضة الثقة تتسرب إلى مهارة، أو أن نجاحاً واحداً يتحول إلى policy عامة، أو أن حلقة لا تعرف متى تتوقف، أو أن ذاكرتها تلخص الفشل بطريقة تمحوه.

من هنا تأتي فرضية هذه الورقة: **الوحدة العلمية المناسبة للأنظمة الوكيلية ليست النموذج وحده ولا الوكيل وحده، بل دورة قدرة محكومة تبدأ بفرق قابل للقياس وتنتهي بأثر عالمي وذاكرة قابلة لإعادة البناء.** هذه الوحدة تمنع مساواة الاستدلال الطويل بالاستدلال الجيد، ومساواة كثرة الوكلاء بالتنوع، ومساواة حفظ السجل بالتعلم، ومساواة وجود أداة بقدرة تشغيلها بأمان.

السؤال الحاكم هو: ما القوانين التي تجعل HyperKnowledge وHyperSkill وHyperAbility وHyperLoop والبروتوكولات فوق المعرفية طبقات نظام تشغيل واحدة؟ نقترح خمسة متغيرات حاكمة: تكاثف الفرق، نفاذية المعرفة، رنين المهارة والحلقة، الجهد السببي للحساب، وصدق إعادة البناء. هذه المتغيرات لا تُعامل كحقائق مثبتة؛ بل كنماذج يجب معايرتها على مهام حقيقية.

تتضمن معايير النجاح ثلاثة مستويات: نجاح task-level، نجاح process-level، ونجاح system-evolution. قد ينجح الوكيل في مهمة ويَفشل النظام لأن المسار غير قابل للتكرار أو لأنه شوّه الذاكرة. وقد يفشل في مهمة لكن ينجح نظامياً إذا حفظ سبب الفشل وضيّق مساحة القرار التالية. لذلك يُقاس النضج بقدرة المنظومة على تحسين احتمال النجاح المتحقق مع خفض المخاطر والهدر والانجراف عبر الزمن.

تفكيك الفجوة إلى اختبارات

تظهر الفجوة في ثلاثة اختبارات عملية. الأول: أعط النظام عشرين trace ناجحاً ثم اسأله ما الذي يستحق التحويل إلى skill؛ إن كانت الإجابة مجرد أكثر المسارات تكراراً فقد غاب الفرق. الثاني: أدخل معلومة صحيحة لكنها غير صالحة للذاكرة العامة؛ إن وصلت إلى policy فقد غابت الحاكمية. الثالث: اجعل مهمة طويلة تحتوي خطوة جانبية صعبة لا تؤثر في الهدف؛ إن استنزف النظام عليها معظم الحساب فقد خلط الصعوبة بالقيمة السببية.

هذه الاختبارات تمنح الورقة حدوداً تشغيلية من البداية. كل نظرية لا تغير قراراً في واحد منها يجب أن تبرهن قيمة أخرى أو تُحذف. والهدف هو منع architecture inflation: تضخم طبقات لا ينتج قرارات مختلفة ولا أثراً أفضل.

2. الخلفية الحديثة: الذاكرة، المهارة، التقييم، وتخصيص حساب وقت الاستدلال

تُظهر أعمال 2026 اتجاهاً واضحاً نحو نقل الاهتمام من «هل يستطيع النموذج الحل؟» إلى «كيف تُدار القدرة أثناء التشغيل؟». إطار SSGM لحوكمة الذاكرة المتطورة يفصل تطور الذاكرة عن حوكمتها ويعالج مخاطر الانجراف والتسمم والتحكم في الوصول. ويقترح MemSkill تحويل عمليات الذاكرة نفسها إلى مهارات قابلة للتعلم والتطور، بما يجعل الإدارة جزءاً من القدرة وليس مجرد مكوّن تخزين.

في المقابل يركز AgentProcessBench على جودة الخطوات الوسيطة في trajectories الأدوات، لأن النجاح النهائي لا يكشف أين ظهرت خطوة محايدة أو خاطئة أو ضارة. هذه النقلة مهمة لفرضية HyperLoop: الحلقة لا ينبغي تقييمها بنتيجتها النهائية فقط، بل بتقدمها المحلي، وبقابلية إصلاحها قبل فعل غير قابل للعكس. أما ATLAS فيجعل تخصيص حساب وقت الاستدلال نفسه قراراً وكيلياً، أي أن orchestrator يمكن أن يقرر متى يستكشف، ومتى يتوقف، وكيف يوزع solver effort.

وتذهب أعمال الذاكرة القائمة على filesystem إلى أن تنظيم الذاكرة ليس مجرد اختيار تمثيل؛ الأدوات، قوة وكيل الإدارة، وحجم التدفق تغير شكل المخزن وكلفة البحث وصحته. هذا يدعم الحاجة إلى «نفاذية معرفية» بدلاً من افتراض أن أي شيء محفوظ يمكن أن ينتقل بحرية إلى policy أو skill. كما أن أعمال co-evolution بين الذاكرة والمهارة تجعل السؤال عن شروط التحويل من trace إلى skill سؤالاً مباشراً لا مجازياً.

النتيجة التي نبني عليها ليست أن النظريات المقترحة غير مسبوقة في مكوناتها، بل أن هناك مساحة تركيبية بين هذه الاتجاهات: **قانون لترقية الفرق إلى قدرة، قانون لعبور المعرفة، قانون لارتباط المهارة بالحلقة، قانون لتوجيه الحساب نحو الأثر، وقانون يمنع إعادة البناء من تزوير التاريخ.**

صلة الأدبيات بالفجوة المقترحة

الأعمال الحديثة توفر مكونات مهمة لكنها لا تحسم تلقائياً علاقات الانتقال بين المكونات. حوكمة الذاكرة تفسر لماذا نحتاج gate، وتطور المهارات يبرر دراسة التحويل من التجربة إلى routine، وتقييم الخطوات يتيح قياس loop dynamics، وagentic test-time scaling يبرر اعتبار compute قراراً. المقترح هنا هو جعل هذه العلاقات نفسها موضوعاً للقياس: ليس فقط ما إذا كانت الذاكرة جيدة، بل ما إذا كان claim مسموحاً له أن يصبح policy؛ وليس فقط ما إذا كانت skill نافعة، بل متى تتكاثف ومتى تقترن بحلقة.

هذا التحديد يسمح بمقارنة عادلة مع الأدبيات: يمكن لكل نظرية أن تفشل حتى لو بقيت المكونات الأساسية مفيدة. ولذلك لا يعتمد البرنامج على الادعاء بأن الحقل يفتقر تماماً إلى مفهوم بعينه، بل على أن مجموعة قرارات الانتقال ما زالت قابلة لصياغة أشد صرامة.

3. النسب المعماري: MCD وQ-MCD وHAKIM-MCD وKB-as-OS

تنطلق المعمارية من MCD بوصفه heuristic لترتيب ما يستحق حساباً وتحقيقاً: الفرق D، التكامل I، النقل T، الوكالة المحدودة A، القوة العملية/التنبؤية P، والتصحيح C، ناقص الضجيج N. لا يهدف MCD إلى ادعاء مقياس نفسي للذكاء، وظيفته هندسية: منع صرف الموارد بالتساوي على فروق غير متساوية.

يضيف Q-MCD طبقة السؤال: قبل أن نبحث عن جواب أفضل نسأل هل السؤال نفسه يفتح الفروق الصحيحة؟ وقد يكون إصلاح السؤال أعلى قيمة من زيادة طول الاستدلال. ثم يأتي HAKIM-MCD كمسار تشغيل: استخراج الفرق، ترتيب المرشحين، توزيع compute، فحص الدليل، الفعل المقيد، التفنيد، قياس الأثر، وإعادة بناء الذاكرة.

داخل KB-as-OS لا تُخزن «المعرفة» كمقاطع فقط. كل وحدة ناضجة تحتاج: تعريفاً، رتبة معرفية، provenance، علاقات تعارض، سياسة استرجاع، أدوات متاحة، محاكم، اختبارات، صلاحيات، شروط ترقية، وذاكرة أثر. HyperSkill هي عقد إجرائي قابل للاستدعاء، HyperAbility تركيب لقدرات جزئية ضمن envelope أداء، HyperLoop آلة حالات محدودة، والبروتوكول فوق المعرفي يقرر متى ينتقل النظام من السؤال إلى البحث إلى الفعل إلى المحكمة.

نقطة الورقة الجديدة هي إضافة طبقة قوانين انتقال بين هذه الوحدات. فبدون قوانين انتقال تصبح architecture registry ضخمة. ومعها يمكن السؤال بدقة: متى يتكاثف الفرق إلى skill؟ متى يسمح للمعرفة أن تعبر إلى policy؟ متى يكون اقتران skill-loop جيداً؟ متى يستحق route حساباً إضافياً؟ ومتى يُسمح لذاكرة جديدة بأن تغير state النظام؟

مثال على المسار الحاكم

في مهمة إصلاح نظام برمجي، قد يكون الفرق الحاكم ليس «اكتب كوداً أفضل» بل «هل الفشل في منطق التطبيق أم في البيئة؟». Q-MCD يرفع هذا السؤال، MCD يعطي أولوية للفرق لأنه يغير route بالكامل، والتروية الحاسوبية تخصص budget أولاً لتجربة تشخيصية. إذا ثبت أن البيئة هي السبب، فلا ينبغي أن تتكاثف تعديلات الكود الفاشلة إلى skill. World-Effect هنا هو نجاح الاختبار في البيئة، والذاكرة تحفظ سبب التمييز لا النص الكامل للمحاولات.

هذا المثال يوضح أن إطار P5 لا يستبدل MCD بل يضيف إليه شروط انتقال: ما الذي يعبر إلى الذاكرة، أي route يستحق compute، وما الذي يستحق الترقية إلى قدرة.

4. منهج توليد النظرية ومحكمة الأصالة

استخدمت الورقة منهجاً تركيبياً من ست مراحل. أولاً: تحديد فجوة آلية لا مجرد فجوة تسمية. ثانياً: بناء baseline صريح يصف ما يحدث إذا لم توجد النظرية. ثالثاً: صياغة متغير أو علاقة يمكن قياسها مباشرة أو بواسطة proxy معلن. رابعاً: اشتقاق أداة أو بروتوكول يجعل النظرية قابلة للتشغيل. خامساً: تصميم تجربة قد ترفضها. سادساً: تشغيل محكمة هدم تسأل ما إذا كان المصطلح الجديد مجرد إعادة وصف لمفهوم معروف.

محكمة الأصالة هنا لا تمنح شهادة «لم يسبق لها مثيل». بدلاً من ذلك تستخدم أربع درجات: (أ) إعادة تسمية؛ (ب) تركيب مفاهيم قائم دون آلية جديدة؛ (ج) تركيب آلي يولد تنبؤاً/قراراً جديداً؛ (د) آلية مدعومة بتجارب مقارنة مستقلة. النظريات الخمس تبدأ في الدرجة (ج) بوصفها مقترحات formalizable، ولا ترتقي إلى (د) إلا بعد تجارب.

تستخدم الورقة كذلك مبدأ الفرق الحاكم: إذا أمكن حذف النظرية مع بقاء القرارات نفسها في كل الحالات، فهي زائدة. وإذا غيرت النظرية قرار التوجيه أو الترقية أو التراجع في حالات محددة، صار لها أثر معماري قابل للاختبار. ومن أجل منع «النظرية التي تفسر كل شيء»، لكل نظرية نطاق صلاحية، متغيرات فشل، وحدود تطبيق.

معيار قبول أداة ناشئة هو أن تختصر قراراً كان يحتاج تفسيراً بشرياً فضفاضاً إلى عقد قابل لإعادة اللعب. ومعيار قبول بروتوكول ناشئ هو أن يحدد states، stop conditions، evidence requirements، transitions، وrollback. بهذه الشروط تصبح الجِدّة أقل اعتماداً على البلاغة وأكثر اعتماداً على الفرق القابل للقياس.

محكمة الجِدّة المضادة للتسمية

لكل مفهوم جديد نطبق اختبار الحذف: احذف اسمه ومعادلته من النظام، ثم حاول إعادة إنتاج القرارات نفسها باستخدام المقاييس الموجودة. إذا أمكن ذلك بلا خسارة، فالمفهوم مجرد alias. ثم اختبار التنبؤ: صمم حالتين متشابهتين ظاهرياً يفترض أن تعطيهما النظرية قرارات مختلفة. وأخيراً اختبار التنفيذ: هل يمكن كتابة دالة أو schema أو protocol يجسد الفرق؟ هذه المحاكم تجعل «الجِدّة» خاصية سلوكية لا انطباعاً لغوياً.

كما يجب إجراء مسح أدبيات أوسع قبل أي claim تاريخي قوي. لذلك تعتمد الورقة استخدام تعبير «مقترحة هنا» بدلاً من «أول نظرية في العالم»، وهو فصل ضروري بين الإبداع التركيبي والإثبات التاريخي للأولوية.

5. النظرية الأولى — تكاثف الفروق الوكيلية ADC

الفكرة: لا ينبغي تعريف «المهارة» بأنها prompt ناجح أو trace محفوظ. نقترح أن المهارة تظهر عندما تتكرر تسوية فرق فعال واحد أو عائلة فروق متشابهة عبر سياقات مختلفة، وتنتج تلك التسوية أثراً متحققاً، ثم يمكن ضغط المسارات المتعددة إلى policy قصيرة مع بقاء الأداء على holdout.

الصيغة الأولية: $ADC = \Sigma(\Delta W \times Transfer \times Evidence \times Novelty) / (Cost + Risk + Redundancy + Drift)$. ΔW هو مقدار التغير العالمي أو التغير في قرار صحيح يمكن نسبه للمسار، Transfer يقيس بقاء الأثر خارج أمثلة البناء، Evidence يزن جودة الإثبات، Novelty هنا ليست أصالة تاريخية بل مقدار المعلومات الإجرائية غير المكررة. المقام يعاقب الكلفة والمخاطر والتكرار والانجراف.

التنبؤ الرئيسي: إذا كانت ADC عالية، فإن استبدال traces الخام بعقد skill مكثف يجب أن يخفض كلفة السياق مع الحفاظ على أو تحسين success rate على عائلات مهام جديدة. إذا انخفض الأداء بقوة بعد الضغط، لم يحدث تكاثف؛ بل حدث اختزال ضار.

يختلف هذا التصور عن حفظ الخبرة لأن «التكاثف» يتطلب محكمة نقل. ويختلف عن مجرد distillation لأن الوحدة لا تُضغط بحسب تشابه النص، بل بحسب الفرق الذي جرى حله والأثر الذي نتج. لذلك قد تتكاثف مسارات لغوياً مختلفة إلى skill واحدة إذا حلت الفرق نفسه، وقد تبقى مسارات متشابهة منفصلة إذا كانت آلياتها السببية مختلفة.

شرط الفشل: إذا أمكن لمقياس ADC المرتفع أن يتنبأ باستمرار بمهارات لا تنقل، أو إذا لم يتفوق على baseline يعتمد فقط على frequency/success، تُخفض النظرية أو تعدل أوزانها.

حالة حدية وتفسير سببي

افترض أن ثلاثة traces حققت النتيجة نفسها: أحدها اعتمد على API، والثاني على تصحيح ملف محلي، والثالث على استنتاج يدوي. التشابه في outcome لا يكفي للتكاثف. DCC يبحث عن invariant transformation؛ قد يكتشف أن الآلية المشتركة هي «تحقق من المصدر قبل إعادة الكتابة» رغم اختلاف الأدوات. إذا لم يوجد invariant سوى النتيجة، تُرفض عملية التكاثف لأن skill ستكون شعاراً لا إجراءً.

ويجب أن تحتفظ candidate skill بأمثلة سلبية: متى لا تستخدم؟ إذا كانت قاعدة الاختيار واسعة أكثر من evidence، ترتفع Redundancy/Drift penalty ويهبط ADC. بذلك تصبح حدود skill جزءاً من المعرفة الإجرائية وليست ملاحظة هامشية.

6. أداة ADC وبروتوكول Δ→Skill وتجارب التفنيد

Difference Condensation Compiler — DCC

يدخل DCC مجموعة trajectories مرتبطة بمهمة أو عدة مهام، لكنه لا يبدأ بالتلخيص. يبدأ باستخراج الفرق الفعال لكل trajectory، ثم يربط كل فرق بالقرار الذي تغير، والدليل الذي برره، والأثر العالمي، وحالة الفشل. بعد ذلك يبني clusters على مستوى «التحويل الإجرائي» لا على مستوى التشابه اللفظي.

بروتوكول Δ→Skill: (1) اجمع ثلاث نجاحات على الأقل من عائلات مهام متباينة؛ (2) استخرج invariant transformation؛ (3) احذف التفاصيل غير اللازمة؛ (4) ولّد عقد skill يحوي trigger/inputs/outputs/preconditions/stop/rollback؛ (5) اختبر holdout؛ (6) شغّل محكمة التلوث؛ (7) قارن الكلفة والأثر قبل/بعد؛ (8) رقِّ إلى HyperSkill فقط إذا حافظت الوحدة على الفرق الحاسم دون trace الأصلي.

تجربة A1: baseline يسترجع أفضل trace خام. النظام المقترح يستخدم skill مكثف. نقيس transfer، latency، tokens، success، وerror type.
تجربة A2: ablation يحذف عنصر الفرق ويضغط بالملخص النصي فقط. إذا لم يتدهور الأداء مقارنة بـDCC، فادعاء مركزية الفرق ضعيف. **تجربة A3:** انتقال cross-domain لاختبار هل التكاثف يلتقط آلية عامة أم pattern محلياً.

المحكمة الأخطر هي «الترقية من نجاح واحد». DCC يجب أن يرفض إنشاء skill دائمة عندما تكون الأدلة قليلة، وأن ينتج بدلاً منها candidate skill بحالة PROPOSED. بهذا تصبح عدم المعرفة حالة تشغيلية شرعية.

سجل DCC المقترح

يحفظ DCC لكل مرشح: world_effects، rejected_traces، supporting_traces، invariant_transformation، active_difference، verifier verdicts، compression_ratio، failure_boundaries، transfer_domains. ويُمنع حذف traces المعارضة؛ تُستخدم لتحديد حدود applicability. عند الترقية، لا يُخزن كل المسار في prompt، بل يُحتفظ برابط provenance يتيح إعادة الاختبار.

الهدف من هذا التصميم هو تحويل «التعلم من الخبرة» إلى عملية قابلة للمحاسبة: يمكن السؤال لاحقاً لماذا توجد skill، ما الأدلة التي أنشأتها، وما الذي سيؤدي إلى downgrade.

7. النظرية الثانية — النفاذية المعرفية المحكومة GEP

الفكرة: معظم بنى الذاكرة تتعامل مع القراءة والكتابة كأفعال. لكن الخطر الأكبر يظهر عند الانتقال بين طبقات ذات صلاحيات معرفية مختلفة: من ملاحظة إلى حقيقة، من ذاكرة episodic إلى policy، من claim غير مؤكد إلى trigger مهارة. نقترح تمثيل هذا الانتقال بمعامل نفاذية Π.

Π = Evidence × Provenance × Reversibility × Coherence / (Drift + Ambiguity + Sensitivity). لا يعني Π احتمالاً بايزياً بالضرورة؛ يمكن معايرته كمؤشر قرار. الدليل والمصدر والقابلية للتراجع والتماسك ترفع النفاذية. الانجراف والغموض والحساسية يخفضانها. معرفة حساسة قد تكون صحيحة جداً لكن نفاذيتها إلى ذاكرة عامة يجب أن تكون منخفضة بسبب نطاق الصلاحية.

تنبؤ النظرية: الأنظمة التي تفصل طبقات الذاكرة لكن تسمح بانتقال غير محكوم ستظهر فيها أخطاء persistent أكثر من أنظمة تستخدم gate معرفياً، حتى إذا كانت جودة الاسترجاع المتجهي متساوية. كما تتنبأ بأن رفع دقة retrieval وحده لا يعالج contamination إذا كان الخطأ في policy التي تسمح بتحويل retrieved claim إلى قاعدة.

النفاذية ليست censorship للمعلومة؛ هي سياسة انتقال. قد يُسمح لـclaim ضعيف بأن يعبر إلى workspace كمقترح، ويُمنع من العبور إلى skill registry. وقد تعبر معلومة قوية إلى قرار محلي ولا تعبر إلى ذاكرة طويلة الأمد إذا كانت خصوصيتها أو تاريخ صلاحيتها يجعل التخزين غير مناسب.

يفشل التصور إذا لم يضف Π قيمة تنبؤية فوق source quality وconfidence التقليديين، أو إذا جعل gates النظام شديد الجمود بحيث يتدهور التعلم المستمر بصورة أكبر من مكاسب السلامة.

طبقات نفاذ مختلفة

يمكن تعريف thresholds متباينة: workspace يقبل proposed claims بسرعة؛ project memory يحتاج provenance واضحاً؛ procedural memory يحتاج تحققاً ونطاق صلاحية؛ skill registry يحتاج نقل وholdout؛ policy عالية الأثر تحتاج مراجعة إضافية. بهذا لا يصبح Π رقماً واحداً ثابتاً بل علاقة بين العنصر والوجهة. نفس المعلومة قد تكون نافذة إلى طبقة وممنوعة عن أخرى.

هذه الصياغة مهمة للخصوصية أيضاً. حقيقة صحيحة عن مستخدم لا ينبغي أن تنتشر لمجرد فائدتها. الحساسية والغرض والصلاحية جزء من denominator أو من gate مستقل fail-closed.

8. EPG وEpistemic Handshake واختبارات التلوث

Epistemic Permeability Gate — EPG

EPG هو وسيط بين طبقات: ability registry، skill registry، procedural memory، declarative memory، working memory، ingestion، policy. كل عبور يحمل «جوازاً معرفياً» يتضمن contradiction links، verifier state، sensitivity، temporal scope، provenance، claim type، rollback path، وdestination policy.

Epistemic Handshake: المصدر يعلن ما يعرفه وما لا يعرفه؛ الوجهة تعلن الحد الأدنى لقبولها؛ gate يحسب Π ويقرر ALLOW / ALLOW-AS-PROPOSED / QUARANTINE / DENY. لا يُسمح للوجهة برفع رتبة العنصر تلقائياً. ويجب أن تحتفظ الذاكرة برابط إلى النسخة السابقة حتى إذا أصبح العنصر superseded.

تجربة B1: حقن claims خاطئة عالية التشابه في memory stream وقياس وصولها إلى policy. **B2:** إدخال claims صحيحة لكنها حساسة واختبار منع تسربها إلى سياق غير مصرح. **B3:** تغيير التاريخ أو المصدر واختبار decay وإعادة التحقق. **B4:** ablation يحذف Reversibility من Π لقياس هل وجود rollback يحسن قرار النفاذ فعلاً أم هو قيد زائد.

المقياس المقترح هو *Propagation Harm per Accepted Item* إضافة إلى false-denial rate. المطلوب ليس منع كل خطأ، بل خفض مقدار الضرر المتراكم من الخطأ الذي يعبر طبقات متعددة. وهذا يتصل مباشرة بمحكمة الذاكرة وبحوكمة المعرفة في M-HKOS.

حالة سباق بين السرعة والسلامة

عند ضغط زمني قد يسمح EPG بمرور claim إلى working state بوسم UNVERIFIED إذا كان الفعل التالي reversible ولا يعرّض بيانات أو أموالاً أو سمعة للخطر. لكن نفس claim لا يمر إلى long-term policy. هكذا لا تعني الحاكمية الانتظار الدائم، بل ربط درجة الإثبات بدرجة الأثر.

للتقييم يجب تسجيل false-accept وfalse-deny معاً. gate يمنع كل شيء ليس جيداً؛ المطلوب frontier بين التعلم والصلابة، ويمكن معايرته حسب المجال.

9. النظرية الثالثة — رنين المهارة-الحلقة SLR

الفكرة: المهارة والحلقة غالباً تصممان منفصلتين: skill تنفذ loop، transformation يكرر أو ينسق الخطوات. نقترح أن جودة النظام تعتمد على اقتران دينامي بينهما. HyperSkill جيدة داخل حلقة سيئة قد تسبب الدوران، وحلقة جيدة مع skill غير قابلة للنقل قد تكرر الفشل بفعالية.

نعرف الرنين الأولي: $\rho = \text{Closure} \times \text{Transfer} \times \text{VerifierAgreement} \times \text{Stability} / (\text{Latency} + \text{Oscillation} + \text{Drift})$. Closure يقيس نسبة الإغلاق المتحقق إلى الإغلاق المقصود، Transfer يقيس أداء skill داخل loops/tasks جديدة، VerifierAgreement يقيس اتفاق تحقق مستقل، Stability يقيس تباين الأداء عبر الزمن. Oscillation يعاقب الانتقال المتكرر بين states دون تقدم.

التنبؤ: أزواج skill-loop ذات ρ أعلى يجب أن تظهر failure recovery أفضل وكلفة أقل من اختيار skill والloop كل على حدة حسب score مستقل. كما نتنبأ النظرية بوجود «anti-resonance»: مهارة ممتازة في loop منخفضة التوافق قد تخفض النظام أكثر من مهارة أبسط.

تضيف النظرية مفهوم *Resonance Frontier*: ليست كل مهارة عامة يجب أن تعمل مع كل loop. لكل skill مجموعة loops تُظهر فيها رنيناً كافياً. وبذلك تصبح القدرة composition graph بدلاً من قائمة skills.

حد الفشل: إذا كان ρ مجرد إعادة تجميع لمقاييس success/latency ولا يحسن اختيار الأزواج على holdout، لا مبرر له. ويجب الحذر من جعل verifier agreement عالياً بسبب تبعية المحققين لنفس النموذج أو البيانات.

مثال anti-resonance

قد تكون skill «ابحث عن مصادر متعددة» ممتازة، لكنها داخل loop محدود بثلاث دقائق قد تسبب تضخم retrieval وتأخير الفعل. في المقابل skill أبسط «تحقق من مصدر رسمي واحد ثم نفذ اختباراً» قد تملك ρ أعلى في سياق التشغيل. هذا يوضح أن جودة المكوّن ليست خاصية مطلقة، هي خاصية اقتران مع dynamics الحلقة.

كما يمكن أن يتغير الرنين مع الزمن إذا تغير API أو النموذج أو verifier. لذلك RLS يحتاج telemetry مستمرة لا score ثابتاً في registry.

10. RLS وResonance Promotion وبروتوكولات كسر الدوران

Resonance Loop Spectrometer — RLS

RLS يبني مصفوفة skill × loop من التنفيذات السابقة ويحسب ρ تحت شروط سياقية: نوع المهمة، مزود النموذج، الأدوات، حدود الوقت، الحساسية، ونمط verifier. لا ينتج score واحداً فقط؛ ينتج profile يكشف أين يفشل الاقتران ولماذا.

Resonance Promotion Protocol: (1) شغّل skill في حلقتين على الأقل؛ (2) احسب closure وoscillation؛ (3) بدّل verifier لاختبار الاستقلال؛ (4) نفذ perturbation للميزانية؛ (5) قس transfer؛ (6) رقِّ pair إلى «resonant pair» إذا تجاوز الحدود؛ (7) عند هبوط ρ عبر نافذة زمنية، خفّض pair أو أعد المعايرة.

نقترح كذلك **Oscillation Breaker**: إذا تكررت state signature بلا خفض في عدم اليقين أو تقدم عالمي، توقف الحلقة وتعيد صياغة الفرق الحاكم بدلاً من المزيد من المحاولات. هذا يمنع illusion of work حيث كثرة الخطوات تبدو نشاطاً بينما لا يتغير العالم.

تجربة C1: مقارنة router يختار أعلى skill score وأعلى loop score منفصلين مقابل router يختار أعلى ρ. **C2:** أدخل verifier correlated ثم مستقل لقياس حساسية الرنين إلى التحقق الزائف. **C3:** نقل skills بين domains لتحديد Resonance Frontier.

تشخيص الطيف

بدلاً من رقم ρ فقط، ينتج RLS أسباب الانخفاض: closure ضعيف، transfer منخفض، oscillation، verifier disagreement، أو latency. هذا يسمح بإصلاح الحلقة الصحيحة. إذا كان الانخفاض بسبب verifier disagreement، تغيير skill قد يكون خطأ؛ نحتاج تحقيقاً. وإذا كان بسبب oscillation، ربما يجب تغيير transition policy لا الأداة.

يُحفظ لكل pair منحنى استقرار عبر versions حتى لا تُرقّى علاقة كانت قوية في بيئة قديمة فقط.

11. النظرية الرابعة — الجهد السببي الحاسوبي CCP

الفكرة: test-time compute يُخصص غالباً بحسب صعوبة متوقعة أو confidence. لكن قد تكون مسألة صعبة عديمة الأثر، ومسألة صغيرة تملك نقطة قرار حرجة. نقترح «الجهد السببي» Φc كمقدار يقدر القيمة الحدية لوحدة compute على تغيير World-Effect مهم مع خفض عدم اليقين.

Φc = E[ΔWorld|route] × Criticality × UncertaintyReduction / (Cost × Risk × Irreversibility). المسار الذي يملك احتمالاً عالياً لتغيير حالة مهمة ويخفض عدم اليقين يستحق حساباً. لكن المخاطر وعدم القابلية للتراجع ترفع الضريبة. يمكن للمعادلة أن توجه إلى verifier أو tool بدلاً من solver أقوى إذا كان ذلك يملك Φc أعلى.

تتصل النظرية بـComputational Vascularity: الحساب يشبه التروية، لكن التروية هنا تُضبط بضغط سببي لا بضغط تعقيد فقط. فإذا كان bottleneck هو معلومة يمكن لأداة موثوقة جلبها، فإن زيادة reasoning tokens قد تملك Φc منخفضة مقارنة باستدعاء الأداة.

التنبؤ: router يخصص budget بحسب Φc سيحقق World-Effect per cost أعلى من fixed budget وdifficulty-only routing في المهام المركبة. كما يتنبأ بأن بعض حالات «reasoning harder» يجب أن تتوقف مبكراً عندما تصبح marginal causal potential سالبة.

حدود النظرية: E[ΔWorld] صعب التقدير قبل الفعل وقد ينحاز نحو أهداف سهلة القياس. لذلك يجب فصل value model عن verifier، واستخدام معايرة لاحقة تقارن التوقع بالأثر الحقيقي.

فرق بين التعقيد والجدوى السببية

قد يواجه الوكيل سؤالاً بحثياً مركباً وتاريخ انتهاء صلاحية API. الأول صعب لغوياً، لكن الثاني قد يحسم نجاح التنفيذ كله. Φc يلتقط أن استدعاء metadata tool قد يكون أعلى قيمة من استدلال طويل. وهكذا يتحول «أين نفكر أكثر؟» إلى «أين يمكن لوحدة compute أن تغير مسار العالم أو تقلل خطراً؟».

يجب أن يتعلم النظام من خطأ التقدير: إذا توقع route أثراً كبيراً ولم ينتج شيئاً، تخفض معايرته في ظروف مشابهة. بهذا يصبح router ذا ذاكرة أداء لا مجرد heuristic ساكن.

12. CPR وCausal Budget Auction واختبارات اقتصاد الاستدلال

Causal Potential Router — CPR

CPR يستقبل candidate routes: reasoning، retrieval، tool call، specialist، verifier، simulation، human escalation. لكل route يقدّر Φc مع نطاق ثقة. ثم يخصص budget كـauction: المسارات تتنافس على compute ليس بقوة الخطاب، بل بالفرق الذي تدعي أنها ستغيره، وبالدليل المتوقع، وبقابلية الرجوع.

Causal Budget Auction: (1) أعلن الفرق الحاكم؛ (2) ولّد routes؛ (3) لكل risk، uncertainty reduction، route: predicted ΔWorld، cost، irreversibility؛ (4) خصص دفعة صغيرة للroutes الأعلى؛ (5) راقب marginal gain؛ (6) أوقف المسار إذا انخفض Φc؛ (7) أعد توزيع الميزانية؛ (8) احفظ prediction-vs-actual لتعلم router.

تجربة D1: fixed tokens مقابل confidence routing مقابل Φc routing. نقيس task success وworld-effect وcost وlatency. **D2:** مهام تحتوي distractor صعباً لكنه غير حاسم لاختبار هل CPR يتجنب إنفاق الحساب عليه. **D3:** مهام عالية المخاطر لاختبار عقوبة irreversibility. **D4:** calibration curve بين Φc المتوقع والأثر الفعلي.

الأداة لا تهدف إلى جعل كل قرار حسابياً دقيقاً، بل إلى فرض سؤال هندسي قبل التوسع: «ما التغير المتوقع مقابل وحدة الكلفة والمخاطر؟». هذا وحده يمكن أن يحد من تضخم test-time compute غير المبرر.

التخصيص المتدرج

الـauction لا يمنح ميزانية كاملة لمسار واحد منذ البداية. يمنح micro-budgets ثم يعيد الحساب بعد observation. هذه البنية تقلل تكلفة التنبؤ الخاطئ وتسمح بالاستكشاف. ويمكن إضافة reservation لمسار verifier بحيث لا يستهلك solver كل الميزانية قبل التحقق.

في المهام عالية الأثر يجب أن تكون Irreversibility penalty غير خطية: خطوة إرسال أو حذف قد تتطلب تحققاً إضافياً حتى لو بدت Φc مرتفعة، لأن خسارة خطأ واحد تتجاوز مكسب السرعة.

13. النظرية الخامسة — إعادة البناء الذاتي المضاد للتاريخ الزائف CSR

الفكرة: الذاكرة المستمرة قد تصبح أكثر خطورة عندما «تتعلم» من تاريخها، لأن التلخيص المتكرر يستطيع محو الفشل أو إعادة تفسير قرار قديم على ضوء نتيجة لاحقة. نقترح CSR: أي تحديث يغير state طويل الأمد يجب أن يثبت أنه متسق مع evidence ledger، وأن ينجح في replay تاريخي ومضاد للواقع.

CSR = HistoricalConsistency + PredictiveGain + ReplayPass - EvidenceLoss - IrreversibilityPenalty. HistoricalConsistency يسأل هل delta الجديدة تتفق مع ما كان معروفاً وقت القرار، لا ما نعرفه الآن فقط. PredictiveGain يقيس هل الذاكرة الجديدة تحسن قراراً لاحقاً. EvidenceLoss يعاقب حذف تفاصيل لازمة لفهم لماذا فشل مسار سابق.

التنبؤ: memory system يستخدم CSR سيقلل hindsight rewriting وcontradiction loss مقارنة بتلخيص تراكمي عادي، وسيحسن reproducibility في جلسات طويلة حتى لو احتاج مساحة أكبر قليلاً.

تقدم النظرية distinction بين «التصحيح» و«إعادة كتابة الماضي». يجوز أن تقول الذاكرة إن claim سابقاً صار superseded، لكن لا يجوز أن تدعي أنه لم يكن موجوداً أو أنه كان معروف الخطأ حينها ما لم يوجد دليل. لذلك تحتفظ CSR بـhistorical states وروابط supersession.

حد الفشل: إذا لم يحسن replay أو القرار المستقبلي، أو إذا أصبحت كلفته أعلى من المكسب بصورة ممنهجة، يجب استخدامه فقط للذاكرة عالية الأثر بدلاً من كل trace.

الفرق بين الذاكرة الملخصة والذاكرة القابلة للمحاكمة

تلخيص «الحل نجح بعد تعديل الإعداد» قد يمحو أن ثلاث محاولات سبقت النجاح وأن أحدها فشل بسبب صلاحية ناقصة. CSR تخزن الحد الأدنى الذي يسمح بإعادة تفسير النجاح: ما كان معروفاً، ما تغير، وما الدليل. هذا مفيد خصوصاً عندما يعاد استخدام الذاكرة لتوليد skills؛ فإذا اختفى سبب الفشل قد تتعلم skill خاطئة.

ولا تتطلب CSR حفظ كل token. يمكن استخدام evidence refs، hashes، checkpoints، وdecision records موجزة، شرط أن تكفي لإعادة بناء الحجة العملية.

14. CRC وHistorical Twin Replay وبروتوكول إعادة البناء

Counterfactual Reconstruction Court — CRC

يستقبل CRC: state_before، evidence_available_then، action، observed outcome، proposed memory delta. ثم يبني «توأماً تاريخياً» يعيد القرار بما كان متاحاً وقتها، وتوأماً مضاداً يغير معلومة أو route حاسمة. إذا كانت delta الجديدة تمحو سبب القرار أو تجعل failure غير قابل للتفسير، تُرفض.

Historical Twin Replay: (1) جمّد timestamp؛ (2) أعد تحميل الأدلة المتاحة حينها؛ (3) استبعد المعرفة اللاحقة؛ (4) أعد تشغيل decision policy؛ (5) قارن القرار؛ (6) شغّل counterfactual route؛ (7) قس ما إذا كانت الذاكرة الجديدة تحافظ على causal attribution؛ (8) خزّن delta مع روابط قبل/بعد.

تجربة E1: توليد جلسات طويلة تحتوي claims تتغير بمرور الزمن؛ قياس historical consistency. **E2:** تكرار summarization 20 دورة وقياس evidence loss. **E3:** إدخال failure ثم success لاحق لاختبار hindsight bias. **E4:** مقارنة append-only ledger + CSR مقابل summary-only memory.

تتوافق CSR مع مبدأ أن الذاكرة ليست «القصة الأفضل» بل بنية إعادة بناء. والنتيجة المرجوة ليست حفظ كل شيء؛ بل حفظ أقل مجموعة أدلة تكفي لإعادة تفسير القرار بطريقة لا تكذب على الماضي.

اختبار التوأم التاريخي

إذا اقترحت الذاكرة الجديدة أن قراراً سابقاً كان «واضح الخطأ»، يعيد CRC تشغيل policy بما كان متاحاً وقتها. إذا كان القرار معقولاً حينها، تُعدل الصياغة إلى «صار superseded بعد وصول دليل جديد». هذا الفارق صغير لغوياً لكنه كبير معرفياً، لأنه يمنع النظام من تضخيم ثقته بتاريخ تعلّمه.

يمكن استخدام replay أيضاً في debugging: إذا لم يعد القرار قابلاً للإنتاج من state المحفوظة، فهناك evidence loss أو تغيّر غير موثق في policy.

15. المعمارية الموحدة P5 فوق HAKIM-MCD

تجمع النظريات الخمس في طبقة P5 فوق المسار المساعفي. يبدأ النظام بـ**السؤال الحاكم** ثم يستخرج active difference ويحسب MCD/Q-MCD. بعد ذلك لا يذهب مباشرة إلى execution. أولاً يمر candidate knowledge عبر GEP؛ ثم يقدر CPR الجهد السببي لكل route؛ ثم تُنفذ HyperSkills داخل HyperLoops وتُقاس علاقاتها بواسطة SLR؛ وعند تكرار تسويات ناجحة يعالج DCC ما إذا كانت تصلح لتكاثف ADC؛ وأخيراً لا تدخل نتائج التطور إلى الذاكرة الطويلة إلا بعد CRC.

المسار المختصر: **Question → Difference → Evidence → Π Gate → Φc Routing → Skill/Ability → Bounded Loop → Courts → World Effect → ρ Analysis → ADC Candidate → CSR Memory.**

يوفر هذا المسار خمسة «صمامات» تمنع تضخم النظام: صمام المعرفة، صمام الحساب، صمام الاقتران، صمام الترقية، وصمام الذاكرة. الفكرة ليست تشغيل جميع الصمامات بأقصى كلفة لكل مهمة. Sparse HyperRouting يحدد أقل صمام لازم حسب الخطر والأثر. مهمة بسيطة قد تمر بسؤال وskill فقط؛ تغيير policy طويل الأمد يحتاج السلسلة الكاملة.

مقياس الصحة العام يمكن صياغته ككثافة قدرة متحققة: Verified transferable world-effect divided by compute + risk + memory drift. هذا لا يثبت AGI؛ لكنه يحول النقاش من «كم طبقة لدينا؟» إلى «كم قدرة قابلة للنقل نحصل عليها لكل كلفة مع بقاء الذاكرة صادقة؟».

سياسة التفعيل المتناثر

لا تعمل P5 نحط أنابيب ثقيل دائماً. مهمة قراءة بسيطة قد تستخدم GEP فقط؛ task متكرر قد يضيف DCC؛ مهمة طويلة تستخدم SLR/CPR؛ تعديل ذاكرة استراتيجية يضيف CRC. router يحدد الطبقات المطلوبة من risk × novelty × expected effect. هذا يحافظ على قيمة النظام دون تحويل الحاكمية إلى عبء ثابت.

ويجب تسجيل سبب عدم تفعيل طبقة، لأن عدم الاستخدام قرار معماري أيضاً. هذا يساعد لاحقاً في معرفة هل failures نتجت عن نقص gate أم عن ضعف مكوّن مفعل.

16. HyperKnowledge-as-OS تحت النظريات الخمس

في هذه المعمارية HyperKnowledge ليست corpus كبيراً فقط. كل Knowledge Unit تحمل هوية، تعريفاً، contradictions، evidence، claims، tests، retrievers، destination permissions، temporal scope، وmemory lineage. ويضيف GEP خاصية جديدة: **مصفوفة نفاذ** تحدد إلى أي طبقات يمكن للوحدة الانتقال وبأي رتبة.

يضيف ADC خاصية «قابلية التكاثف»: هل هذه المعرفة مجرد وصف أم أنها تشارك في تحويل متكرر يمكن أن يصبح skill؟ ويضيف CCP خاصية «الجهد السببي»: هل استرجاع هذه الوحدة سيغير route أو قراراً؟ وهنا يتجاوز retrieval التشابه إلى relevance السببي.

يمكن تصميم index مركب: semantic similarity + active-difference match + evidence rank + causal potential + freshness + privacy scope. ويصبح GraphRAG ليس فقط شبكة entities، بل شبكة claims→mechanisms→skills→loops→effects→memory deltas. عندها يمكن للنظام الإجابة: أي معرفة أنتجت أي قدرة؟ وأي أثر يبرر بقاء claim في الطبقة التشغيلية؟

التقييم لا يكتفي بـRecall@K. نقترح: Causal Retrieval Gain، Permeability Error، Evidence Precision، Difference Recall، Contradiction Preservation، وContext Economy. المعرفة الجيدة هي ما يصل في الوقت الصحيح، بالرتبة الصحيحة، إلى الوجهة الصحيحة، ويؤدي إلى قرار أفضل دون تسريب أو تلوث.

وحدة معرفة تشغيلية نموذجية

قد تحمل وحدة عن «صلاحيات أداة الملفات» claim موثقاً، تاريخ صلاحية، test، destination policy، contradictions يثبت السلوك، وlinks إلى skills تستخدمها. عند retrieval لا يُسأل فقط هل النص مشابه، بل هل الفرق الحالي يتعلق بالصلاحيات؟ هل المعلومة ما زالت نافذة؟ هل لها Φc كافٍ لتغيير route؟ هكذا يتحول index من ذاكرة نصية إلى substrate للقرار.

وتُعد جودة الضغط مهمة: إذا أعادت وحدة صغيرة كل القرارات الحاسمة التي كان corpus الكبير ينتجها، ارتفعت كثافة المعرفة التشغيلية.

17. HyperSkill وHyperAbility كطبقتي تشغيل

HyperSkill عقد محدد: failure modes، rollback، stop، metrics، verifier، outputs، transformation، preconditions، inputs، trigger، transfer boundary. وتحت ADC لا تصبح skill «ناضجة» إلا عندما يمكن إثبات أنها ضغطت traces متعددة دون فقد الآلية الحاسمة. تحت SLR لا يكفي أن تنجح skill منفردة؛ يجب معرفة loops التي ترن معها.

HyperAbility أعلى مستوى: مجموعة skills/loops تحقق وظيفة مركبة ضمن envelope. نقترح تمثيل القدرة كـgraph لا كقائمة. العقد skills؛ الحواف resonance profiles؛ والمسارات abilities. كل ability تحمل maturity: PROPOSED → TESTED → TRANSFERRED → GOVERNED → DEPLOYABLE، مع downgrade ممكن.

يساعد GEP على منع انتقال معرفة غير ناضجة إلى شروط skill. ويساعد CCP على اختيار هل نحتاج skill جديدة أم verifier أو tool. ويساعد CSR على منع ability registry من نسيان الحالات التي فشلت فيها القدرة عند تعريف حدودها.

مقاييس القدرة: Safety Envelope، Cost-Adjusted Reliability، World-Effect Closure، Failure Recovery، Transfer Frontier، وDrift Rate. ولا يجوز استخدام benchmark واحد كدليل على قدرة عامة. المطلوب طيف عائلات مهام، تغير الأدوات، تغير السياق، ومفاجآت failure.

مخطط ترقية القدرة

تنتقل HyperAbility من PROPOSED إلى TESTED بعد اختبارات داخل المجال، ثم TRANSFERRED بعد نجاح عائلات جديدة، ثم GOVERNED بعد اكتمال حدود الصلاحية والrollback، ثم DEPLOYABLE بعد مراقبة زمنية. أي regression يمكن أن يعيدها إلى مرحلة أدنى. هذا السلم يمنع كلمة «ability» من التحول إلى ادعاء نهائي.

كما يجب فصل capability coverage عن permission scope: نظام قادر تقنياً على فعل شيء لا يعني أنه مخول لتنفيذه.

18. HyperLoop وHyperCognitive Protocols

HyperLoop هي آلة حالات وليست «كرر حتى تنجح». أقل عقد للحلقة: state transitions، progress metric، budget، goal state، initial state، trace schema، rollback، escalation، stop، evidence conditions. ويضيف SLR قدرة على معرفة هل المهارة المختارة تناسب dynamics الحلقة. ويضيف CPR budget reallocation أثناء التشغيل.

البروتوكولات فوق المعرفية تحكم متى يغير النظام نمط عمله. مثال FATTĀH: Frame ثم Atomize ثم Triangulate ثم Test ثم Act ثم Harvest. ويمكن ربطه بـP5: في Frame يُحدد الفرق؛ في Triangulate يمر الدليل عبر GEP؛ في Test يوزع CPR الحساب؛ في Act تقاس SLR؛ في Harvest يُختبر ADC وCSR.

نقترح Protocol Stack: Governing Question Protocol، Difference Extraction، Epistemic Handshake، Causal Budget Auction، Historical Twin Replay، World-Effect Verification، Skill-Loop Resonance. كل protocol يملك entry/exit condition ولا يعمل كـprompt زخرفي.

تظهر قيمة هذه الطبقة في الفشل. إذا فشل tool، لا يكرر النظام بشكل أعمى. يسأل: هل تغير الفرق؟ هل فقدت المعرفة نفاذيتها؟ هل انخفض Φc للمسار؟ هل ظهر anti-resonance؟ هل يجب أن يحفظ failure كمرشح ADC سلبي؟ بهذه الطريقة يصبح الفشل معلومة تشغيلية.

قانون عدم التقدم

نقترح شرطاً عاماً: إذا لم ينخفض عدم اليقين ولم يتغير state ولم يزد evidence خلال نافذة محددة، فالحلقة لا تتقدم حتى لو زادت الرسائل. عندها إما يعاد تعريف الفرق، أو يغير route، أو يعلن BLOCKED. هذا القانون يترجم anti-loop rhetoric إلى إشارة قابلة للقياس.

يصبح البروتوكول فوق المعرفي مسؤولاً عن الانتقال بين modes، بينما تبقى المهارة مسؤولة عن transformation محلي؛ الفصل يمنع skill من السيطرة على الهدف العام.

19. الأدوات الخمس الناشئة وواجهاتها

الأداة	المدخل	القرار	المخرج
DCC	trajectories + effects	هل يوجد تكاثف فرق؟	candidate HyperSkill + evidence
EPG	claim passport + destination	هل تعبر المعرفة؟ وبأي رتبة؟	ALLOW / PROPOSED / QUARANTINE / DENY
RLS	skill × loop executions	هل يوجد رنين أم anti-resonance؟	ρ profile + compatible frontier
CPR	candidate routes + budgets	أين توجد أعلى قيمة سببية هامشية؟	budget allocation + stop
CRC	before/after states + evidence	هل Memory Delta صادقة تاريخياً؟	ACCEPT / REVISE / REJECT

واجهة موحدة مقترحة لكل أداة: `{task_id, governing_question, active_difference, inputs, evidence_refs, authority_scope, budget, output_schema, verifier, rollback}`. وتعيد كل أداة: `{decision, confidence_band, evidence_used, unresolved, failure_modes, metric_delta, next_action, memory_residue}`.

تفرض هذه الواجهة مبدأين: الأدوات لا تملك حق تغيير الهدف الحاكم، ولا تكتب الذاكرة الطويلة مباشرة. هي تقترح Memory Residue يمر عبر CRC/EPG حسب وجهته. بذلك تُفصل القدرة عن السلطة، والتوليد عن الاعتماد.

يمكن تنفيذ الأدوات أولاً كـServices Python/TypeScript أو prompts مهيكلة، ثم استبدال بعض heuristics بنماذج تعلمية بعد تجميع بيانات. المهم أن يبقى contract ثابتاً بما يسمح بالمقارنة والrollback.

قابلية الاستبدال

يمكن تنفيذ كل أداة heuristically أولاً ثم استبدالها بنموذج متعلم من دون تغيير contract. هذه خاصية مهمة للاستمرارية: EPG rule-based اليوم قد يصبح classifier معياراً غداً، وCPR heuristic قد يصبح learned allocator، لكن سجل المدخل والقرار والrollback يبقى ثابتاً. وهكذا لا يرتبط OS بنموذج أو مزود واحد.

20. الخوارزمية الموحدة: من الطلب إلى الأثر والذاكرة

```
INPUT task, context, authority
Q  := governing_question(task)
Δ  := extract_active_difference(Q, context)
E  := retrieve_and_rank_evidence(Δ)
E' := EPG(E, destination="working_state")
routes := propose_routes(Δ, E')
for r in routes: r.phi := CPR.estimate(r)
while budget_available:
    r := argmax_marginal_phi(routes)
    result := execute_bounded(r)
    step_verdict := process_court(result)
    update_phi_from_observation(r)
    if world_effect_verified or stop_condition: break
rho := RLS(skill, loop, trajectory)
adc := DCC.consider(trajectory_family)
delta := propose_memory_delta(result, courts, effect)
final_delta := CRC(delta, historical_state)
write_if_permitted(EPG(final_delta, destination="long_term_memory"))
```

هذه الخوارزمية لا تشترط استخدام كل مكوّن. route policy يختار الحد الأدنى الكافي. لكن ترتيبها يحمي من اختصارات شائعة: لا skill قبل دليل نقل، لا memory write قبل محكمة، لا compute expansion قبل تقدير قيمة هامشية، ولا world-effect بلا صلاحية.

نقطة مهمة هي فصل **decision trace** عن chain-of-thought الخاصة بالنموذج. المطلوب سجل موجز: ما الفرق؟ ما الأدلة؟ ما route؟ ما الأداة؟ ما الحكم؟ ما الأثر؟ هذا يكفي للتدقيق والتعلم ولا يتطلب تخزين استدلال داخلي مطول.

سياسة الفشل

أي خطوة قد تعيد واحدة من أربع حالات: VERIFIED، PARTIAL، BLOCKED، FAILED. لا يُجبر النظام على نتيجة ناجحة. عند BLOCKED تحفظ المعلومة الناقصة والسبب والroute الذي جُرب، وعند FAILED تُمنع الترقية ويُقترح تعديل. هذه الحالات تجعل المسار قابلاً للاستئناف وتمنع اختلاق النجاح لاستكمال pipeline.

21. برنامج التجارب والمعايير المقترحة

نقترح ست عائلات تجريبية. **Family 1 — Skill Distillation:** مهام برمجية/بحثية/تشغيلية متكررة لاختبار ADC ضد trace retrieval وprompt templates. **Family 2 — Memory Contamination:** streams تحوي claims خاطئة أو حساسة أو قديمة لاختبار GEP. **Family 3 — Long-Horizon Loops:** tool tasks مع recovery ودوران لاختبار SLR. **Family 4 — Compute Allocation:** مسائل تحوي distractors وصناديق أدوات لاختبار CCP. **Family 5 — Historical Integrity:** جلسات طويلة متغيرة الحقائق لاختبار CSR. **Family 6 — Full Stack:** مهام end-to-end تختبر تفاعل النظريات.

المقاييس الأساسية: transfer score، irreversible error count، latency، tokens، verified world effect، step effectiveness، task success، rollback success، memory contradiction preservation، oscillation count، contamination propagation. تضاف المقاييس الجديدة: ADC calibration، CSR replay pass، Φc، ρ، Π، score.

يجب استخدام baselines قوية وablations: حذف active difference، حذف provenance، اختيار skill/loop مستقلاً، routing بالconfidence فقط، summary-only memory. ويجب تنفيذ holdout على domains مختلفة لمنع overfitting المعماري.

التجارب لا ينبغي أن تقيّم النظام بواسطة judge واحد من العائلة نفسها فقط. حيث يمكن تُستخدم اختبارات حتمية؛ وحيث لا يمكن تستخدم لجنة مقيمين مع قياس الاستقلال والتباين. وأي نتيجة موجبة بلا artifacts قابلة لإعادة التشغيل تبقى «reported» لا «verified».

تصميم إحصائي وتقييم زمني

يفضل استخدام repeated runs لأن الوكلاء stochastic، والإبلاغ عن المتوسط والتباين لا أفضل محاولة فقط. كما يجب فصل development tasks عن holdout، وقياس الأداء بعد أسابيع أو تغييرات أدوات لاختبار drift. ولCSR نحتاج تقييم longitudinal؛ ولSLR نحتاج matrices عبر عدة loops؛ ولCCP نحتاج calibration لا accuracy فقط.

إشارة مهمة هي «كلفة الخطأ غير القابل للعكس» التي يجب تسجيلها منفصلة عن task failure، لأن نظاماً يحقق success أعلى لكنه يرسل أفعالاً خاطئة أكثر قد يكون أسوأ عملياً.

22. محكمة الهدم: اعتراضات قد تُسقط الإطار

الاعتراض 1: خمسة مقاييس مركبة تعيد تسمية أشياء معروفة. الرد ليس بلاغياً: إذا لم تغير المقاييس قرارات التوجيه/الترقية مقارنة بالbaselines، تُحذف. **الاعتراض 2: كثرة gates ترفع latency.** لذلك Sparse HyperRouting يجعل P5 risk-adaptive. **الاعتراض 3: تقدير Φc غير موثوق.** يجب معايرته prediction-vs-actual واستخدامه كترتيب لا كحقيقة.

الاعتراض 4: ADC قد يضغط تفاصيل حاسمة. holdout وanti-reduction court شرطان. **الاعتراض 5: Π قد يخنق التعلم.** نحتاج stability–plasticity tuning وALLOW-AS-PROPOSED بدلاً من binary deny. **الاعتراض 6: ρ قد يكافئ loops قصيرة فقط.** يجب تطبيع closure بحسب صعوبة وهدف الحلقة. **الاعتراض 7: CSR قد يصبح مكلفاً جداً.** يُفعّل بكثافة أعلى للذاكرة التي تغير policy أو صلاحية.

الاعتراض 8: النظام يخلق illusion of AGI عبر بنية كبيرة. هذا اعتراض قاتل إذا تحولت الأسماء إلى بديل عن القياس. لذلك الورقة تحصر الدعوى في «AGI-oriented architecture» وتطلب أثر ونقل. **الاعتراض 9: المعادلات heuristic.** صحيح؛ هي formal handles أولية تحتاج معايرة، وليست قوانين طبيعية.

الحكم الحالي: الإطار يصلح كبرنامج بحثي قابل للتنفيذ، لا كنظرية مثبتة. قوته في أنه يجعل نقاط الانتقال بين المعرفة والقدرة والحساب والذاكرة موضوعاً للتجربة بدلاً من تركها افتراضات ضمنية.

معايير الرفض المسبق

قبل التجارب نعلن شروط الرفض لتقليل تحيز التأويل: ADC تُرفض إن لم تحسن transfer/cost؛ GEP تُرفض إن لم تخفض propagation harm عند false-deny مقبول؛ SLR تُرفض إن لم تتنبأ بالأزواج أفضل من scores المستقلة؛ CCP تُرفض إن لم تحسن world-effect per cost؛ CSR تُرفض إن لم تحسن historical replay أو reproducibility. ويمكن بقاء بعض النظريات وسقوط أخرى من دون إسقاط الإطار كله.

24. الخاتمة والمراجع المختارة

خارطة التنفيذ المختصرة: ابدأ بعقود JSON/YAML للمعرفة والمهارة والقدرة والحلقة والأثر والذاكرة، ثم نفّذ DCC/EPG/RLS/CPR/CRC كقواعد قابلة للتفسير، اجمع telemetry على مهام حقيقية، عاير الأوزان على holdout، وبعدها فقط اسمح بأجزاء متعلمة مع versioning وrollback. لا يبدأ البرنامج ببناء آلاف المهارات؛ يبدأ بعشر مهام متكررة وأدلة قابلة لإعادة التشغيل.

قدمت الورقة خمس فرضيات آلية لتقوية حلقة الانتقال من المعرفة إلى القدرة ثم إلى الفعل والذاكرة. ADC يختبر متى يصبح حل الفرق مهارة؛ GEP يضبط عبور المعرفة؛ SLR يربط المهارة بديناميات الحلقة؛ CCP يوجه compute نحو الأثر السببي؛ وCSR يحمي الذاكرة من إعادة كتابة التاريخ. معاً تشكل P5 طبقة حاكمية قابلة للدمج مع MCD/Q-MCD وHAKIM-MCD وKB-as-OS وHyperSkills وBounded HyperLoops وWorld-Effect وSelf-Reconstruction Memory.

المعيار النهائي ليس عدد النظريات ولا حجم الحزمة، بل قابلية التفنيد. إذا لم تتفوق أداة على baseline، تُخفض. إذا لم ينقل skill، لا تُرقّى. إذا لم يغير compute الأثر، يُوقف. إذا لم تحفظ الذاكرة التاريخ، تُرفض delta. بهذه الصرامة يمكن لحزمة AGI-oriented أن تكون مختبراً هندسياً بدلاً من معجم أسماء.

مراجع مختارة

1. Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.
2. Schick et al. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761.
3. Lightman et al. Let's Verify Step by Step. arXiv:2305.20050.
4. Packer et al. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.
5. Rafailov et al. Direct Preference Optimization. arXiv:2305.18290.
6. Snell et al. Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Model Parameters. arXiv:2408.03314.
7. Xie et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. arXiv:2404.07972.
8. Lam et al. Governing Evolving Memory in LLM Agents: SSGM Framework. arXiv:2603.11768.
9. Zhang et al. MemSkill: Learning and Evolving Memory Skills for Self-Evolving Agents. arXiv:2602.02474.
10. AgentProcessBench: Diagnosing Step-Level Process Quality in Tool-Using Agents. arXiv:2603.14465.
11. Qin, Cao, Xie. ATLAS: Agentic Test-time Learning-to-Allocate Scaling. arXiv:2606.01667.
12. Tang et al. From Memory to Skills: Evidence-Grounded Co-Evolution Governance for Long-Horizon LLM Agents. arXiv:2607.16621.
13. Zhou et al. Filesystem-Based Memory for LLM Agents: Organization, Evolution, and Sustainability. arXiv:2607.26637.
14. Moussaif, Z. HKOS-M–MICAGFI: من النموذج اللغوي إلى النظام المعرفي الوكيلي المحكوم. corpus داخلي، 2026.
15. Moussaif, Z. Moussaif AGI Model Training Operating System — 600K Lines. corpus داخلي، 2026.

الحالة: برنامج بحثي تأسيسي. الخطوة العلمية التالية هي تنفيذ benchmark harness وإصدار نتائج قابلة لإعادة الإنتاج، لا توسيع الأسماء قبل القياس.