الذكاء الاصطناعي المعرفي - الجزء الثاني

نظريات ومفاهيم جديدة لتطوير المجال رياضياً وتقنياً وبرمجياً وهندسياً وتدريبياً

هذا الامتداد يبدأ من آخر صفحة في النسخة الأولى، ويضيف 120 صفحة بحثية كثيفة. هدفه ليس إعادة شرح ما سبق، بل تحويله إلى برنامج منافسة: مفاهيم قابلة للتنفيذ، نماذج رياضية، هندسة وكلاء، بروتوكولات تدريب، معايير تقييم، وخطة مختبر يمكن أن ينافس الشركات الكبرى عبر السرعة المفاهيمية لا عبر الإنفاق وحده.

التمييز المنهجي: ما يرد عن الأوراق والمصادر الحديثة يُعامل كمعطى موثّق، وما يرد في النماذج الجديدة يُعامل كفرضيات أو تركيبات قابلة للهدم. كل نظرية في هذه الصفحات ستُربط بآلية، ومعادلة، ونواة تنفيذ، وبروتوكول اختبار.

مبدأ المنافسة: الشركة الكبيرة تفوز بالحوسبة والبيانات؛ المختبر الصغير يمكن أن يفوز بالبنية: ذاكرة خبرية، تحقق سببي، وكلاء متخصصون، تجارب سريعة، وسجل أخطاء يتحول إلى أصول تدريبية.

امتداد بعد الصفحة 33 من النسخة الأولى - صفحة افتتاح الجزء الثاني.

2. مصفوفة المصادر التي تغير اتجاه الكتاب

هذه المصفوفة لا تُستخدم لتزيين المراجع، بل لتحديد ما يغيّر النموذج. مصادر الوكلاء اللغويين تدفعنا إلى الذاكرة والفعل؛ مصادر نماذج العالم تدفعنا إلى المحاكاة والتجسد؛ مصادر التحقق والتعلم المعزز تدفعنا إلى المدقق والمكافأة؛ مصادر الأنظمة متعددة الوكلاء تدفعنا إلى الشركة كعقل موزع.

الرمز	المصدر	النوع/السنة	أثره في النموذج
S1	CoALA: Cognitive Architectures for Language Agents	Sumers, Yao, Narasimhan, Griffiths, TMLR/ arXiv 2024	ينظم الوكلاء اللغويين حول الذاكرة، فضاء الفعل، واتخاذ القرار.
S2	A Survey on the Evolution of LLM Agent Memory Mechanisms	Luo et al., arXiv 2026	يقترح تطور ذاكرة الوكلاء من التخزين إلى التأمل ثم الخبرة.
S3	Memory for Autonomous LLM Agents	Du, arXiv 2026	يؤطر الذاكرة كحلقة write-manage-read مرتبطة بالإدراك والفعل.
S4	A-MEM: Agentic Memory for LLM Agents	Xu et al., arXiv/OpenReview 2025	يبني ذاكرة وكيلية قابلة للتنظيم والتكيّف وتقليل كلفة السياق.
S5	A Path Towards Autonomous Machine Intelligence	LeCun, OpenReview 2022	يقترح نماذج عالمية تنبؤية وحافزاً داخلياً وتخطيطاً هرمياً.
S6	Comprehensive Survey on World Models for Embodied AI	Li et al., arXiv 2025	يصنف نماذج العالم في الذكاء المتجسد حسب الوظيفة والزمن والمكان.
S7	Embodied AI Agents: Modeling the World	Fung et al., arXiv 2025	يربط الوكلاء المتجسدين بنماذج العالم والذاكرة وفهم نوايا المستخدم.
S8	A Survey on Test-Time Scaling in LLMs	Zhang et al., arXiv 2025	يقسم توسيع الحوسبة وقت الاختبار إلى ماذا وكيف وأين وكيف نقيس.
S9	Process Reward Models That Think	Khalifa et al., arXiv 2025	يعالج المدقق/النموذج المكافئ ككائن يفكر قبل إعطاء الدرجة.

قاعدة قراءة: لا تُقرأ المصادر كحقائق مكتملة، بل كقيود تصميم. المصدر الجيد لا يقول لنا فقط ماذا نفعل؛ يقول لنا أين لم ينجح الآخرون بعد.

امتداد بعد الصفحة 33 من النسخة الأولى - صفحة 2 من 120.

3. مصادر حديثة للمرحلة التالية

الرمز	المصدر	النوع/السنة	أثره في النموذج
S10	Reward Reasoning Model	Guo et al., arXiv 2025	يقترح نماذج مكافأة تمارس تفكيراً مقصوداً قبل الحكم.
S11	RL for Large Reasoning Models Survey	Zhang et al., arXiv 2025	يراجع التعلم المعزز في تدريب نماذج الاستدلال الكبيرة.
S12	Neuro-Symbolic AI in 2024	Colelough et al., arXiv 2025	يراجع دمج الرمزية والنماذج العصبية، بما فيها LLM والمعماريات.
S13	Multi-Agent Collaboration Mechanisms	Tran et al., arXiv 2025	يصنف التعاون بين الوكلاء حسب الفاعلين والهياكل والبروتوكولات.
S14	MultiAgentBench	Zhu et al., ACL/arXiv 2025	يقيس التعاون والمنافسة في أنظمة الوكلاء متعددة النماذج.
S15	SWE-bench	Princeton/NLP official benchmark	يقيّم وكلاء البرمجة على إصلاح قضايا حقيقية في مستودعات برمجية.
S16	AlphaEvolve	DeepMind / arXiv 2025	وكيل تطوري يطوّر الخوارزميات عبر تعديل الكود وتقييمه.
S17	Agentic AI comprehensive survey	Artificial Intelligence Review 2025	يراجع معمارية الوكلاء العصريين بين الرمزية وLLM والأدوات.

تمثل هذه المصادر خطاً فاصلاً بين الذكاء الاصطناعي كمولد نصوص والذكاء الاصطناعي كنظام معرفي. أهم النقاط: الذاكرة لم تعد مخزناً، صارت إدارة خبرة. الاستدلال لم يعد prompt، صار تدريباً وتوسيعاً وقت الاختبار. العالم لم يعد نصاً، صار نموذجاً داخلياً قابلاً للمحاكاة. الشركة لم تعد فريقاً واحداً، صارت بنية متعددة الوكلاء.

سؤال حاكم: ما البنية الأصغر التي تجعل الوكيل يتعلم من الفشل ويحوّله إلى قاعدة قابلة للاستعمال لاحقاً؟

امتداد بعد الصفحة 33 من النسخة الأولى - صفحة 3 من 120.

4. أفق المنافسة الكبرى: نظرية الحافة المعرفية

الفكرة المركزية. نظرية الحافة المعرفية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نظرية الحافة المعرفية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نظرية الحافة المعرفية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نظرية الحافة المعرفية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة نظرية الحافة المعرفية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نظرية الحافة المعرفية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

5. أفق المنافسة الكبرى: مصفوفة التفوق البحثي

الفكرة المركزية. مصفوفة التفوق البحثي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مصفوفة التفوق البحثي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مصفوفة التفوق البحثي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مصفوفة التفوق البحثي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة مصفوفة التفوق البحثي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مصفوفة التفوق البحثي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

6. أفق المنافسة الكبرى: مختبر الوكيل الممتد

الفكرة المركزية. مختبر الوكيل الممتد ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مختبر الوكيل الممتد باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مختبر الوكيل الممتد هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مختبر الوكيل الممتد عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة مختبر الوكيل الممتد نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مختبر الوكيل الممتد إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

7. أفق المنافسة الكبرى: ميزانية الندرة المعرفية

الفكرة المركزية. ميزانية الندرة المعرفية ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ميزانية الندرة المعرفية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ميزانية الندرة المعرفية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ميزانية الندرة المعرفية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة ميزانية الندرة المعرفية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ميزانية الندرة المعرفية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

8. أفق المنافسة الكبرى: المؤسسة كعقل موزع

الفكرة المركزية. المؤسسة كعقل موزع ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج المؤسسة كعقل موزع باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر المؤسسة كعقل موزع هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس المؤسسة كعقل موزع عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة المؤسسة كعقل موزع نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل المؤسسة كعقل موزع إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

9. أفق المنافسة الكبرى: منهجية الشركة البحثية الصغيرة

الفكرة المركزية. منهجية الشركة البحثية الصغيرة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منهجية الشركة البحثية الصغيرة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منهجية الشركة البحثية الصغيرة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منهجية الشركة البحثية الصغيرة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة منهجية الشركة البحثية الصغيرة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منهجية الشركة البحثية الصغيرة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

10. أفق المنافسة الكبرى: بروتوكول تحويل الورقة إلى منتج

الفكرة المركزية. بروتوكول تحويل الورقة إلى منتج ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بروتوكول تحويل الورقة إلى منتج باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بروتوكول تحويل الورقة إلى منتج هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بروتوكول تحويل الورقة إلى منتج عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة بروتوكول تحويل الورقة إلى منتج نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بروتوكول تحويل الورقة إلى منتج إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

11. أفق المنافسة الكبرى: معايير الفوز على المختبرات الكبرى

الفكرة المركزية. معايير الفوز على المختبرات الكبرى ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معايير الفوز على المختبرات الكبرى باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معايير الفوز على المختبرات الكبرى هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معايير الفوز على المختبرات الكبرى عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة معايير الفوز على المختبرات الكبرى نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معايير الفوز على المختبرات الكبرى إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

12. أفق المنافسة الكبرى: حلقات التسريع المعرفي

الفكرة المركزية. حلقات التسريع المعرفي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج حلقات التسريع المعرفي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر حلقات التسريع المعرفي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس حلقات التسريع المعرفي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة حلقات التسريع المعرفي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل حلقات التسريع المعرفي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

13. أفق المنافسة الكبرى: خارطة دفاع ضد التضخيم

الفكرة المركزية. خارطة دفاع ضد التضخيم ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج خارطة دفاع ضد التضخيم باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر خارطة دفاع ضد التضخيم هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس خارطة دفاع ضد التضخيم عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة خارطة دفاع ضد التضخيم نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل خارطة دفاع ضد التضخيم إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

14. النظام التشغيلي المعرفي: نواة معرفية تنفيذية

الفكرة المركزية. نواة معرفية تنفيذية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نواة معرفية تنفيذية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نواة معرفية تنفيذية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نواة معرفية تنفيذية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة نواة معرفية تنفيذية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نواة معرفية تنفيذية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

15. النظام التشغيلي المعرفي: مجدول أفكار متعدد السرعات

الفكرة المركزية. مجدول أفكار متعدد السرعات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مجدول أفكار متعدد السرعات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مجدول أفكار متعدد السرعات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مجدول أفكار متعدد السرعات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة مجدول أفكار متعدد السرعات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مجدول أفكار متعدد السرعات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

16. النظام التشغيلي المعرفي: متحكم ذاكرة وفعل

الفكرة المركزية. متحكم ذاكرة وفعل ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج متحكم ذاكرة وفعل باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر متحكم ذاكرة وفعل هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس متحكم ذاكرة وفعل عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة متحكم ذاكرة وفعل نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل متحكم ذاكرة وفعل إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

17. النظام التشغيلي المعرفي: طبقة قرار قابلة للتدقيق

الفكرة المركزية. طبقة قرار قابلة للتدقيق ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج طبقة قرار قابلة للتدقيق باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر طبقة قرار قابلة للتدقيق هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس طبقة قرار قابلة للتدقيق عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة طبقة قرار قابلة للتدقيق نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل طبقة قرار قابلة للتدقيق إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

18. النظام التشغيلي المعرفي: نظام ملفات للمعنى

الفكرة المركزية. نظام ملفات للمعنى ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نظام ملفات للمعنى باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نظام ملفات للمعنى هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نظام ملفات للمعنى عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة نظام ملفات للمعنى نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نظام ملفات للمعنى إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

19. النظام التشغيلي المعرفي: واجهة أدوات آمنة

الفكرة المركزية. واجهة أدوات آمنة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج واجهة أدوات آمنة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر واجهة أدوات آمنة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس واجهة أدوات آمنة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة واجهة أدوات آمنة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل واجهة أدوات آمنة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

20. النظام التشغيلي المعرفي: سجل نوايا قابل للمراجعة

الفكرة المركزية. سجل نوايا قابل للمراجعة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج سجل نوايا قابل للمراجعة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر سجل نوايا قابل للمراجعة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس سجل نوايا قابل للمراجعة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة سجل نوايا قابل للمراجعة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل سجل نوايا قابل للمراجعة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

21. النظام التشغيلي المعرفي: محرك التزامات

الفكرة المركزية. محرك التزامات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج محرك التزامات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر محرك التزامات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس محرك التزامات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة محرك التزامات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل محرك التزامات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

22. النظام التشغيلي المعرفي: مراقب تناقضات

الفكرة المركزية. مراقب تناقضات ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مراقب تناقضات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مراقب تناقضات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مراقب تناقضات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة مراقب تناقضات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مراقب تناقضات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

23. النظام التشغيلي المعرفي: نظام أذونات معرفية

الفكرة المركزية. نظام أذونات معرفية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نظام أذونات معرفية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نظام أذونات معرفية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نظام أذونات معرفية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة نظام أذونات معرفية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نظام أذونات معرفية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

24. ذاكرة الخبرة والتراكم: الذاكرة كمنحنى خبرة

الفكرة المركزية. الذاكرة كمنحنى خبرة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج الذاكرة كمنحنى خبرة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر الذاكرة كمنحنى خبرة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس الذاكرة كمنحنى خبرة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة الذاكرة كمنحنى خبرة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل الذاكرة كمنحنى خبرة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

25. ذاكرة الخبرة والتراكم: تحويل المسار إلى قاعدة

الفكرة المركزية. تحويل المسار إلى قاعدة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تحويل المسار إلى قاعدة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تحويل المسار إلى قاعدة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تحويل المسار إلى قاعدة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = \mathrm{Compress}(\mathrm{Trace}) - \mathrm{Forgetting}(\mathrm{Bias}) + \mathrm{Counterfactual}(\mathrm{Trace})$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة تحويل المسار إلى قاعدة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تحويل المسار إلى قاعدة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

26. ذاكرة الخبرة والتراكم: ضغط التجربة دون قتل السياق

الفكرة المركزية. ضغط التجربة دون قتل السياق ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ضغط التجربة دون قتل السياق باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ضغط التجربة دون قتل السياق هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ضغط التجربة دون قتل السياق عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة ضغط التجربة دون قتل السياق نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ضغط التجربة دون قتل السياق إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

27. ذاكرة الخبرة والتراكم: نسيان محسوب

الفكرة المركزية. نسيان محسوب ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نسيان محسوب باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نسيان محسوب هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نسيان محسوب عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة نسيان محسوب نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نسيان محسوب إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

28. ذاكرة الخبرة والتراكم: كشف التناقضات في الذاكرة

الفكرة المركزية. كشف التناقضات في الذاكرة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج كشف التناقضات في الذاكرة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر كشف التناقضات في الذاكرة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس كشف التناقضات في الذاكرة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة كشف التناقضات في الذاكرة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل كشف التناقضات في الذاكرة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

29. ذاكرة الخبرة والتراكم: ذاكرة ذات طبقات زمنية

الفكرة المركزية. ذاكرة ذات طبقات زمنية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذاكرة ذات طبقات زمنية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذاكرة ذات طبقات زمنية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذاكرة ذات طبقات زمنية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i) \propto \exp((Q_i - Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة ذاكرة ذات طبقات زمنية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذاكرة ذات طبقات زمنية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

30. ذاكرة الخبرة والتراكم: ذاكرة اجتماعية متعددة الوكلاء

الفكرة المركزية. ذاكرة اجتماعية متعددة الوكلاء ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذاكرة اجتماعية متعددة الوكلاء باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذاكرة اجتماعية متعددة الوكلاء هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذاكرة اجتماعية متعددة الوكلاء عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة ذاكرة اجتماعية متعددة الوكلاء نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذاكرة اجتماعية متعددة الوكلاء إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

31. ذاكرة الخبرة والتراكم: بنك فشل منتج

الفكرة المركزية. بنك فشل منتج ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بنك فشل منتج باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بنك فشل منتج هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بنك فشل منتج عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة بنك فشل منتج نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بنك فشل منتج إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

32. ذاكرة الخبرة والتراكم: ذاكرة هندسية للكود

الفكرة المركزية. ذاكرة هندسية للكود ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذاكرة هندسية للكود باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذاكرة هندسية للكود هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذاكرة هندسية للكود عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة ذاكرة هندسية للكود نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذاكرة هندسية للكود إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

33. ذاكرة الخبرة والتراكم: ذاكرة بحثية للمختبر

الفكرة المركزية. ذاكرة بحثية للمختبر ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذاكرة بحثية للمختبر باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذاكرة بحثية للمختبر هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذاكرة بحثية للمختبر عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة ذاكرة بحثية للمختبر نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذاكرة بحثية للمختبر إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

34. نماذج العالم والتجسد: عالم داخلي قابل للتشغيل

الفكرة المركزية. عالم داخلي قابل للتشغيل ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج عالم داخلي قابل للتشغيل باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر عالم داخلي قابل للتشغيل هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس عالم داخلي قابل للتشغيل عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة عالم داخلي قابل للتشغيل نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل عالم داخلي قابل للتشغيل إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

امتداد بعد الصفحة 33 من النسخة الأولى - موضع الادعاء: نموذج/فرضية قابلة للاختبار، لا حقيقة نهائية.

35. نماذج العالم والتجسد: محاكي سببي صغير

الفكرة المركزية. محاكي سببي صغير ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج محاكي سببي صغير باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر محاكي سببي صغير هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس محاكي سببي صغير عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة محاكي سببي صغير نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل محاكي سببي صغير إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

36. نماذج العالم والتجسد: تجسد افتراضي قبل الروبوت

الفكرة المركزية. تجسد افتراضي قبل الروبوت ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تجسد افتراضي قبل الروبوت باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تجسد افتراضي قبل الروبوت هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تجسد افتراضي قبل الروبوت عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة تجسد افتراضي قبل الروبوت نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تجسد افتراضي قبل الروبوت إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

37. نماذج العالم والتجسد: توأم رقمي معرفي

الفكرة المركزية. توأم رقمي معرفي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج توأم رقمي معرفي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر توأم رقمي معرفي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس توأم رقمي معرفي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة توأم رقمي معرفي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل توأم رقمي معرفي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

38. نماذج العالم والتجسد: استشعار معنى لا صورة

الفكرة المركزية. استشعار معنى لا صورة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج استشعار معنى لا صورة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر استشعار معنى لا صورة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس استشعار معنى لا صورة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة استشعار معنى لا صورة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل استشعار معنى لا صورة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

39. نماذج العالم والتجسد: تخطيط في الخيال

الفكرة المركزية. تخطيط في الخيال ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تخطيط في الخيال باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تخطيط في الخيال هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تخطيط في الخيال عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة تخطيط في الخيال نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تخطيط في الخيال إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

40. نماذج العالم والتجسد: اتساق فيزيائي طويل المدى

الفكرة المركزية. اتساق فيزيائي طويل المدى ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج اتساق فيزيائي طويل المدى باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر اتساق فيزيائي طويل المدى هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس اتساق فيزيائي طويل المدى عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة اتساق فيزيائي طويل المدى نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل اتساق فيزيائي طويل المدى إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

41. نماذج العالم والتجسد: نمذجة نية المستخدم

الفكرة المركزية. نمذجة نية المستخدم ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نمذجة نية المستخدم باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نمذجة نية المستخدم هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نمذجة نية المستخدم عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة نمذجة نية المستخدم نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نمذجة نية المستخدم إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

42. نماذج العالم والتجسد: عالم اجتماعي داخلي

الفكرة المركزية. عالم اجتماعي داخلي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج عالم اجتماعي داخلي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر عالم اجتماعي داخلي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس عالم اجتماعي داخلي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة عالم اجتماعي داخلي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل عالم اجتماعي داخلي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

43. نماذج العالم والتجسد: موازنة المحاكاة والواقع

الفكرة المركزية. موازنة المحاكاة والواقع ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج موازنة المحاكاة والواقع باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر موازنة المحاكاة والواقع هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس موازنة المحاكاة والواقع عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة موازنة المحاكاة والواقع نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل موازنة المحاكاة والواقع إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

44. السببية والتحقق: مدقق سببي

الفكرة المركزية. مدقق سببي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مدقق سببي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مدقق سببي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مدقق سببي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة مدقق سببي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مدقق سببي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

45. السببية والتحقق: مكافأة تفكر

الفكرة المركزية. مكافأة تفكر ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مكافأة تفكر باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مكافأة تفكر هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مكافأة تفكر عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = \text{Compress}(\text{Trace}) - \text{Forgetting}(\text{Bias}) + \text{Counterfactual}(\text{Trace})$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة مكافأة تفكر نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مكافأة تفكر إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

46. السببية والتحقق: برهان قابل للتنفيذ

الفكرة المركزية. برهان قابل للتنفيذ ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج برهان قابل للتنفيذ باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر برهان قابل للتنفيذ هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس برهان قابل للتنفيذ عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة برهان قابل للتنفيذ نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل برهان قابل للتنفيذ إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

47. السببية والتحقق: التحقق من المسار لا النتيجة

الفكرة المركزية. التحقق من المسار لا النتيجة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج التحقق من المسار لا النتيجة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر التحقق من المسار لا النتيجة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس التحقق من المسار لا النتيجة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة التحقق من المسار لا النتيجة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل التحقق من المسار لا النتيجة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

48. السببية والتحقق: حكم متعدد الوكلاء

الفكرة المركزية. حكم متعدد الوكلاء ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج حكم متعدد الوكلاء باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر حكم متعدد الوكلاء هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس حكم متعدد الوكلاء عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة حكم متعدد الوكلاء نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل حكم متعدد الوكلاء إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

49. السببية والتحقق: استدلال ضدّي

الفكرة المركزية. استدلال ضدّي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج استدلال ضدّي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر استدلال ضدّي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس استدلال ضدّي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة استدلال ضدّي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل استدلال ضدّي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

50. السببية والتحقق: خريطة عيوب سببية

الفكرة المركزية. خريطة عيوب سببية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج خريطة عيوب سببية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر خريطة عيوب سببية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس خريطة عيوب سببية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة خريطة عيوب سببية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل خريطة عيوب سببية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

51. السببية والتحقق: اختبار كسر النموذج

الفكرة المركزية. اختبار كسر النموذج ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج اختبار كسر النموذج باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر اختبار كسر النموذج هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس اختبار كسر النموذج عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة اختبار كسر النموذج نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل اختبار كسر النموذج إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

52. السببية والتحقق: محكمة آلية

الفكرة المركزية. محكمة آلية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج محكمة آلية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر محكمة آلية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس محكمة آلية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة محكمة آلية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل محكمة آلية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

53. السببية والتحقق: هندسة الثقة

الفكرة المركزية. هندسة الثقة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج هندسة الثقة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر هندسة الثقة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس هندسة الثقة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة هندسة الثقة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل هندسة الثقة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

54. تدريب نماذج الاستدلال: منهج تدريب متعدد الأطوار

الفكرة المركزية. منهج تدريب متعدد الأطوار ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منهج تدريب متعدد الأطوار باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منهج تدريب متعدد الأطوار هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منهج تدريب متعدد الأطوار عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة منهج تدريب متعدد الأطوار نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منهج تدريب متعدد الأطوار إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

55. تدريب نماذج الاستدلال: تعلم معزز قابل للتحقق

الفكرة المركزية. تعلم معزز قابل للتحقق ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تعلم معزز قابل للتحقق باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تعلم معزز قابل للتحقق هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تعلم معزز قابل للتحقق عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = \text{Compress}(\text{Trace}) - \text{Forgetting}(\text{Bias}) + \text{Counterfactual}(\text{Trace})$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة تعلم معزز قابل للتحقق نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تعلم معزز قابل للتحقق إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

56. تدريب نماذج الاستدلال: توسيع وقت الاختبار

الفكرة المركزية. توسيع وقت الاختبار ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج توسيع وقت الاختبار باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر توسيع وقت الاختبار هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس توسيع وقت الاختبار عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة توسيع وقت الاختبار نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل توسيع وقت الاختبار إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

57. تدريب نماذج الاستدلال: تدريب المدقق قبل المفكر

الفكرة المركزية. تدريب المدقق قبل المفكر ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تدريب المدقق قبل المفكر باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تدريب المدقق قبل المفكر هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تدريب المدقق قبل المفكر عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^*=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة تدريب المدقق قبل المفكر نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تدريب المدقق قبل المفكر إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

58. تدريب نماذج الاستدلال: تقطير المسارات الناجحة

الفكرة المركزية. تقطير المسارات الناجحة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تقطير المسارات الناجحة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تقطير المسارات الناجحة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تقطير المسارات الناجحة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة تقطير المسارات الناجحة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تقطير المسارات الناجحة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

59. تدريب نماذج الاستدلال: توليد بيانات صعبة

الفكرة المركزية. توليد بيانات صعبة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج توليد بيانات صعبة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر توليد بيانات صعبة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس توليد بيانات صعبة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة توليد بيانات صعبة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل توليد بيانات صعبة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

60. تدريب نماذج الاستدلال: منع حفظ الحلول

الفكرة المركزية. منع حفظ الحلول ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منع حفظ الحلول باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منع حفظ الحلول هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منع حفظ الحلول عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة منع حفظ الحلول نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منع حفظ الحلول إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

61. تدريب نماذج الاستدلال: نمذجة الخطأ

الفكرة المركزية. نمذجة الخطأ ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نمذجة الخطأ باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نمذجة الخطأ هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نمذجة الخطأ عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة نمذجة الخطأ نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نمذجة الخطأ إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

62. تدريب نماذج الاستدلال: تدريب على الرفض الذكي

الفكرة المركزية. تدريب على الرفض الذكي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تدريب على الرفض الذكي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تدريب على الرفض الذكي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تدريب على الرفض الذكي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة تدريب على الرفض الذكي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تدريب على الرفض الذكي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

63. تدريب نماذج الاستدلال: مقاييس تفكير لا مقاييس نص

الفكرة المركزية. مقاييس تفكير لا مقاييس نص ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مقاييس تفكير لا مقاييس نص باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مقاييس تفكير لا مقاييس نص هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تملك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مقاييس تفكير لا مقاييس نص عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة مقاييس تفكير لا مقاييس نص نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مقاييس تفكير لا مقاييس نص إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

64. الأنظمة متعددة الوكلاء: سوق معرفي داخلي

الفكرة المركزية. سوق معرفي داخلي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج سوق معرفي داخلي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر سوق معرفي داخلي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس سوق معرفي داخلي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة سوق معرفي داخلي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل سوق معرفي داخلي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

65. الأنظمة متعددة الوكلاء: مناظرة مقيدة

الفكرة المركزية. مناظرة مقيدة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مناظرة مقيدة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مناظرة مقيدة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مناظرة مقيدة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة مناظرة مقيدة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مناظرة مقيدة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

66. الأنظمة متعددة الوكلاء: تنسيق نجمي وشجري ورسومي

الفكرة المركزية. تنسيق نجمي وشجري ورسومي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تنسيق نجمي وشجري ورسومي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تنسيق نجمي وشجري ورسومي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تنسيق نجمي وشجري ورسومي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة تنسيق نجمي وشجري ورسومي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تنسيق نجمي وشجري ورسومي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

67. الأنظمة متعددة الوكلاء: ذاكرة جمعية

الفكرة المركزية. ذاكرة جمعية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذاكرة جمعية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذاكرة جمعية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذاكرة جمعية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة ذاكرة جمعية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذاكرة جمعية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

68. الأنظمة متعددة الوكلاء: وكيل ناقد ووكلاء منتجون

الفكرة المركزية. وكيل ناقد ووكلاء منتجون ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل ناقد ووكلاء منتجون باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل ناقد ووكلاء منتجون هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل ناقد ووكلاء منتجون عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة وكيل ناقد ووكلاء منتجون نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل ناقد ووكلاء منتجون إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

69. الأنظمة متعددة الوكلاء: اختيار قيادة ديناميكي

الفكرة المركزية. اختيار قيادة ديناميكي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج اختيار قيادة ديناميكي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر اختيار قيادة ديناميكي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس اختيار قيادة ديناميكي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة اختيار قيادة ديناميكي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل اختيار قيادة ديناميكي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

70. الأنظمة متعددة الوكلاء: تخصيص أدوار ذاتي

الفكرة المركزية. تخصيص أدوار ذاتي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تخصيص أدوار ذاتي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تخصيص أدوار ذاتي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تخصيص أدوار ذاتي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة تخصيص أدوار ذاتي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تخصيص أدوار ذاتي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

71. الأنظمة متعددة الوكلاء: منع عدوى الخطأ

الفكرة المركزية. منع عدوى الخطأ ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منع عدوى الخطأ باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منع عدوى الخطأ هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منع عدوى الخطأ عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة منع عدوى الخطأ نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منع عدوى الخطأ إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

72. الأنظمة متعددة الوكلاء: عقد عمل بين الوكلاء

الفكرة المركزية. عقد عمل بين الوكلاء ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج عقد عمل بين الوكلاء باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر عقد عمل بين الوكلاء هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس عقد عمل بين الوكلاء عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة عقد عمل بين الوكلاء نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل عقد عمل بين الوكلاء إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

73. الأنظمة متعددة الوكلاء: ذكاء جماعي قابل للقياس

الفكرة المركزية. ذكاء جماعي قابل للقياس ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ذكاء جماعي قابل للقياس باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ذكاء جماعي قابل للقياس هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ذكاء جماعي قابل للقياس عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة ذكاء جماعي قابل للقياس نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ذكاء جماعي قابل للقياس إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

74. الهندسة البرمجية والتشغيل: مكدس وكيل صناعي

الفكرة المركزية. مكدس وكيل صناعي ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مكدس وكيل صناعي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مكدس وكيل صناعي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مكدس وكيل صناعي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة مكدس وكيل صناعي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مكدس وكيل صناعي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

75. الهندسة البرمجية والتشغيل: خط أنابيب بحث-كود-تقييم

الفكرة المركزية. خط أنابيب بحث-كود-تقييم ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج خط أنابيب بحث-كود-تقييم باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر خط أنابيب بحث-كود-تقييم هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس خط أنابيب بحث-كود-تقييم عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = \mathrm{Compress}(\mathrm{Trace}) - \mathrm{Forgetting}(\mathrm{Bias}) + \mathrm{Counterfactual}(\mathrm{Trace})$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة خط أنابيب بحث-كود-تقييم نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل خط أنابيب بحث-كود-تقييم إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

76. الهندسة البرمجية والتشغيل: منصة تجارب قابلة للإعادة

الفكرة المركزية. منصة تجارب قابلة للإعادة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منصة تجارب قابلة للإعادة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منصة تجارب قابلة للإعادة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منصة تجارب قابلة للإعادة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة منصة تجارب قابلة للإعادة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منصة تجارب قابلة للإعادة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

77. الهندسة البرمجية والتشغيل: حاويات معرفة

الفكرة المركزية. حاويات معرفة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج حاويات معرفة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر حاويات معرفة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس حاويات معرفة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة حاويات معرفة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل حاويات معرفة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

78. الهندسة البرمجية والتشغيل: بوابة أدوات

الفكرة المركزية. بوابة أدوات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بوابة أدوات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بوابة أدوات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بوابة أدوات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة بوابة أدوات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بوابة أدوات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

79. الهندسة البرمجية والتشغيل: مراقبة تكلفة التفكير

الفكرة المركزية. مراقبة تكلفة التفكير ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مراقبة تكلفة التفكير باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مراقبة تكلفة التفكير هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مراقبة تكلفة التفكير عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة مراقبة تكلفة التفكير نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مراقبة تكلفة التفكير إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

80. الهندسة البرمجية والتشغيل: هندسة سجلات

الفكرة المركزية. هندسة سجلات ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج هندسة سجلات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر هندسة سجلات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس هندسة سجلات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة هندسة سجلات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل هندسة سجلات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

81. الهندسة البرمجية والتشغيل: أمان ضد حقن التعليمات

الفكرة المركزية. أمان ضد حقن التعليمات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج أمان ضد حقن التعليمات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر أمان ضد حقن التعليمات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس أمان ضد حقن التعليمات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة أمان ضد حقن التعليمات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل أمان ضد حقن التعليمات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

82. الهندسة البرمجية والتشغيل: اختبار وكلاء البرمجة

الفكرة المركزية. اختبار وكلاء البرمجة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج اختبار وكلاء البرمجة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر اختبار وكلاء البرمجة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس اختبار وكلاء البرمجة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة اختبار وكلاء البرمجة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل اختبار وكلاء البرمجة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

83. الهندسة البرمجية والتشغيل: نشر تدريجي مضبوط

الفكرة المركزية. نشر تدريجي مضبوط ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نشر تدريجي مضبوط باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نشر تدريجي مضبوط هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نشر تدريجي مضبوط عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة نشر تدريجي مضبوط نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نشر تدريجي مضبوط إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

84. الرياضيات والبنية الشكلية: ديناميك الحالة المعرفية

الفكرة المركزية. ديناميك الحالة المعرفية ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج ديناميك الحالة المعرفية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر ديناميك الحالة المعرفية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس ديناميك الحالة المعرفية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة ديناميك الحالة المعرفية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل ديناميك الحالة المعرفية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

85. الرياضيات والبنية الشكلية: دالة قيمة معرفية

الفكرة المركزية. دالة قيمة معرفية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج دالة قيمة معرفية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر دالة قيمة معرفية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس دالة قيمة معرفية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة دالة قيمة معرفية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل دالة قيمة معرفية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

86. الرياضيات والبنية الشكلية: مقياس MCD موسع

الفكرة المركزية. مقياس MCD موسع ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مقياس MCD موسع باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مقياس MCD موسع هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مقياس MCD موسع عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة مقياس MCD موسع نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مقياس MCD موسع إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

87. الرياضيات والبنية الشكلية: معادلة الحافز

الفكرة المركزية. معادلة الحافز ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معادلة الحافز باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معادلة الحافز هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معادلة الحافز عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة معادلة الحافز نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معادلة الحافز إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

88. الرياضيات والبنية الشكلية: قياس الاتساق السببي

الفكرة المركزية. قياس الاتساق السببي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج قياس الاتساق السببي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر قياس الاتساق السببي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس قياس الاتساق السببي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة قياس الاتساق السببي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل قياس الاتساق السببي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

89. الرياضيات والبنية الشكلية: مسافة الذاكرة

الفكرة المركزية. مسافة الذاكرة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مسافة الذاكرة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مسافة الذاكرة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مسافة الذاكرة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة مسافة الذاكرة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مسافة الذاكرة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

90. الرياضيات والبنية الشكلية: طاقة التناقض

الفكرة المركزية. طاقة التناقض ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج طاقة التناقض باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر طاقة التناقض هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس طاقة التناقض عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة طاقة التناقض نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل طاقة التناقض إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

91. الرياضيات والبنية الشكلية: مؤشر الانفتاح

الفكرة المركزية. مؤشر الانفتاح ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مؤشر الانفتاح باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مؤشر الانفتاح هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مؤشر الانفتاح عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة مؤشر الانفتاح نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مؤشر الانفتاح إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

92. الرياضيات والبنية الشكلية: مقياس قابلية النقل

الفكرة المركزية. مقياس قابلية النقل ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مقياس قابلية النقل باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مقياس قابلية النقل هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مقياس قابلية النقل عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة مقياس قابلية النقل نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مقياس قابلية النقل إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

93. الرياضيات والبنية الشكلية: حدود الموارد

الفكرة المركزية. حدود الموارد ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج حدود الموارد باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر حدود الموارد هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس حدود الموارد عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة حدود الموارد نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل حدود الموارد إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

94. التدريب والبيانات والمعايير: بيانات مسارات لا أجوبة

الفكرة المركزية. بيانات مسارات لا أجوبة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بيانات مسارات لا أجوبة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بيانات مسارات لا أجوبة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بيانات مسارات لا أجوبة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة بيانات مسارات لا أجوبة نضيف مقياساً خاصاً: هل تحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بيانات مسارات لا أجوبة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

95. التدريب والبيانات والمعايير: مختبر مهام متدرج

الفكرة المركزية. مختبر مهام متدرج ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مختبر مهام متدرج باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مختبر مهام متدرج هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مختبر مهام متدرج عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة مختبر مهام متدرج نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مختبر مهام متدرج إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

96. التدريب والبيانات والمعايير: معيار ذاكرة طويلة

الفكرة المركزية. معيار ذاكرة طويلة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معيار ذاكرة طويلة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معيار ذاكرة طويلة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معيار ذاكرة طويلة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة معيار ذاكرة طويلة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معيار ذاكرة طويلة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

97. التدريب والبيانات والمعايير: معيار عالم مفتوح

الفكرة المركزية. معيار عالم مفتوح ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معيار عالم مفتوح باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معيار عالم مفتوح هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معيار عالم مفتوح عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة معيار عالم مفتوح نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معيار عالم مفتوح إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

98. التدريب والبيانات والمعايير: معيار تعاون

الفكرة المركزية. معيار تعاون ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معيار تعاون باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معيار تعاون هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معيار تعاون عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة معيار تعاون نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معيار تعاون إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

99. التدريب والبيانات والمعايير: معيار هندسي

الفكرة المركزية. معيار هندسي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج معيار هندسي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر معيار هندسي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس معيار هندسي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة معيار هندسي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل معيار هندسي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

100. التدريب والبيانات والمعايير: منهج كسر متعمد

الفكرة المركزية. منهج كسر متعمد ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منهج كسر متعمد باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منهج كسر متعمد هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منهج كسر متعمد عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة منهج كسر متعمد نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منهج كسر متعمد إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

101. التدريب والبيانات والمعايير: بيانات أخلاقية قابلة للتنفيذ

الفكرة المركزية. بيانات أخلاقية قابلة للتنفيذ ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بيانات أخلاقية قابلة للتنفيذ باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بيانات أخلاقية قابلة للتنفيذ هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بيانات أخلاقية قابلة للتنفيذ عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة بيانات أخلاقية قابلة للتنفيذ نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بيانات أخلاقية قابلة للتنفيذ إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

102. التدريب والبيانات والمعايير: تحليل الانحياز

الفكرة المركزية. تحليل الانحياز ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج تحليل الانحياز باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر تحليل الانحياز هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس تحليل الانحياز عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة تحليل الانحياز نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل تحليل الانحياز إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

103. التدريب والبيانات والمعايير: سجل تجارب علمي

الفكرة المركزية. سجل تجارب علمي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج سجل تجارب علمي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر سجل تجارب علمي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس سجل تجارب علمي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة سجل تجارب علمي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل سجل تجارب علمي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

104. التطبيقات والشركات: وكيل بحث علمي

الفكرة المركزية. وكيل بحث علمي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل بحث علمي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل بحث علمي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل بحث علمي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة وكيل بحث علمي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل بحث علمي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

105. التطبيقات والشركات: وكيل هندسة برمجية

الفكرة المركزية. وكيل هندسة برمجية ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل هندسة برمجية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل هندسة برمجية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل هندسة برمجية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة وكيل هندسة برمجية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل هندسة برمجية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

106. التطبيقات والشركات: وكيل تصميم شرائح

الفكرة المركزية. وكيل تصميم شرائح ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل تصميم شرائح باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل تصميم شرائح هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل تصميم شرائح عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة وكيل تصميم شرائح نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل تصميم شرائح إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

107. التطبيقات والشركات: وكيل طب معرفي مساعد

الفكرة المركزية. وكيل طب معرفي مساعد ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل طب معرفي مساعد باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل طب معرفي مساعد هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل طب معرفي مساعد عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة وكيل طب معرفي مساعد نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل طب معرفي مساعد إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

108. التطبيقات والشركات: وكيل تعليم شخصي

الفكرة المركزية. وكيل تعليم شخصي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل تعليم شخصي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل تعليم شخصي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل تعليم شخصي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة وكيل تعليم شخصي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غير ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل تعليم شخصي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

109. التطبيقات والشركات: وكيل مختبر روبوتي

الفكرة المركزية. وكيل مختبر روبوتي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S1]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل مختبر روبوتي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل مختبر روبوتي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل مختبر روبوتي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة وكيل مختبر روبوتي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل مختبر روبوتي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

110. التطبيقات والشركات: وكيل إدارة معرفة مؤسسة

الفكرة المركزية. وكيل إدارة معرفة مؤسسة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S2]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل إدارة معرفة مؤسسة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل إدارة معرفة مؤسسة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل إدارة معرفة مؤسسة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة وكيل إدارة معرفة مؤسسة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل إدارة معرفة مؤسسة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

111. التطبيقات والشركات: وكيل اكتشاف خوارزميات

الفكرة المركزية. وكيل اكتشاف خوارزميات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S3]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل اكتشاف خوارزميات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل اكتشاف خوارزميات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تملك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل اكتشاف خوارزميات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$s_{t+1}=F_\theta(s_t,o_t,a_t,m_t,g_t)+\epsilon_t$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة وكيل اكتشاف خوارزميات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل اكتشاف خوارزميات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

112. التطبيقات والشركات: وكيل أمن سيبراني

الفكرة المركزية. وكيل أمن سيبراني ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S5]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج وكيل أمن سيبراني باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر وكيل أمن سيبراني هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس وكيل أمن سيبراني عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$J=\mathbb{E}[R]-\lambda C_{compute}-\mu H_{risk}+\beta I_{novelty}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة وكيل أمن سيبراني نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل وكيل أمن سيبراني إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

113. التطبيقات والشركات: منصة Cognitive AI نخدمة

الفكرة المركزية. منصة Cognitive AI نخدمة ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S6]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج منصة Cognitive AI نخدمة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر منصة Cognitive AI نخدمة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس منصة Cognitive AI نخدمة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$M_{t+1}=Consolidate(M_t,\tau_t,\rho_t,\kappa_t)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة منصة Cognitive AI نخدمة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل منصة Cognitive AI نخدمة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء، الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

114. النظرية الكبرى وبرنامج الاختبار: نموذج عقل-عالم-غاية

الفكرة المركزية. نموذج عقل-عالم-غاية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S8]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج نموذج عقل-عالم-غاية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر نموذج عقل-عالم-غاية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس نموذج عقل-عالم-غاية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$V_c(x)=P(consistent\mid x)-P(causal\;break\mid x)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة نموذج عقل-عالم-غاية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل نموذج عقل-عالم-غاية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

115. النظرية الكبرى وبرنامج الاختبار: الحاكمية الحسابية

الفكرة المركزية. الحاكمية الحسابية ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S9]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج الحاكمية الحسابية باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر الحاكمية الحسابية هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس الحاكمية الحسابية عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$\Delta K = Compress(Trace) - Forgetting(Bias) + Counterfactual(Trace)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة الحاكمية الحسابية نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل الحاكمية الحسابية إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

116. النظرية الكبرى وبرنامج الاختبار: التراكب الوعائي كعدسة

الفكرة المركزية. التراكب الوعائي كعدسة ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S11]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج التراكب الوعائي كعدسة باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر التراكب الوعائي كعدسة هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس التراكب الوعائي كعدسة عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$A^*=\arg\max_a\;\mathbb{E}_{w\sim W}[U(s,a,w)]-Cost(a)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
for step in episode:
    obs = perceive(env)
    state = causal_update(state, obs)
    plan = reason(state, memory, goals)
    action = verifier.select(plan.candidates)
    memory.write(trace(obs, plan, action))
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: طبقه في محاكاة عالم مفتوح، أضف أشياء غير معروفة أثناء المهمة وقس قدرة الوكيل على تعديل تمثيلاته دون إعادة تدريب كاملة. في حالة التراكب الوعائي كعدسة نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل التراكب الوعائي كعدسة إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

117. النظرية الكبرى وبرنامج الاختبار: حلقة الوعي العملي

الفكرة المركزية. حلقة الوعي العملي ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S12]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج حلقة الوعي العملي باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر حلقة الوعي العملي هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس حلقة الوعي العملي عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$MCD^{*}=D_{alt}+D_{time}+D_{causal}+D_{agency}+D_{telos}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
def consolidate(memory, trace):
    rule = abstract(trace)
    if verifier(rule) > threshold:
        memory.graph.merge(rule)
    else:
        memory.failures.add(trace)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: استعمل مدققين مستقلين: مدقق نتيجة، مدقق مسار، ومدقق سببي. لا تقبل إجابة نجحت مصادفة إذا فشل مسارها. في حالة حلقة الوعي العملي نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل حلقة الوعي العملي إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

118. النظرية الكبرى وبرنامج الاختبار: مخطط بحث لخمس سنوات

الفكرة المركزية. مخطط بحث لخمس سنوات ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S13]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج مخطط بحث لخمس سنوات باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر مخطط بحث لخمس سنوات هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس مخطط بحث لخمس سنوات عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً، إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$F_{active}=E_q[\log q(z)-\log p(o,z)]$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
class CognitiveKernel:
    def decide(self, obs):
        z = self.world.encode(obs)
        causes = self.causal.infer(z)
        return self.policy.search(causes, self.memory)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: دربه بمنهج curriculum يبدأ بمهام قصيرة ثم يرفع طول الحلقة وعدد الأدوات وعدد القيود الأخلاقية. في حالة مخطط بحث لخمس سنوات نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل مخطط بحث لخمس سنوات إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

119. النظرية الكبرى وبرنامج الاختبار: بروتوكول نشر أوراق

الفكرة المركزية. بروتوكول نشر أوراق ليس شعاراً، هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S15]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج بروتوكول نشر أوراق باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر بروتوكول نشر أوراق هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس بروتوكول نشر أوراق عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$P_{select}(r_i)\propto \exp((Q_i-Risk_i)/T)$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
reward = outcome_score(y)
reward += alpha * process_verifier(steps)
reward -= beta * unsafe_action_penalty(trace)
update(policy, reward)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اجعل كل فشل ينتج عينة تدريب جديدة: سبب الفشل، القرار البديل، القاعدة المستخلصة، وقيمة المخاطرة. في حالة بروتوكول نشر أوراق نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل بروتوكول نشر أوراق إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.

120. النظرية الكبرى وبرنامج الاختبار: خطة بناء فريق

الفكرة المركزية. خطة بناء فريق ليس شعاراً؛ هو وحدة بناء في برنامج بحثي يريد نقل الذكاء الاصطناعي المعرفي من مستوى النماذج اللغوية المعزولة إلى مستوى أنظمة معرفية قابلة للتشغيل. المنطلق هو أن النظام الذكي لا يفوز بمجرد أن يجيب، بل حين يملك دورة كاملة: يدرك، يكوّن حالة داخلية، يسترجع خبرة، يضع خطة، يختبرها، ينفذ، ثم يعيد كتابة ذاكرته. هذا المعنى قريب من موجة الوكلاء اللغويين، لكنه يتطلب ضبطاً هندسياً ورياضياً أعمق من مجرد سلسلة prompts. الإحالة الخلفية لهذه الصفحة هي [S16]، مع اعتبار ما يلي تركيباً جديداً مقترحاً لا حقيقة مثبتة.

القاعدة النظرية. يجب أن نعالج خطة بناء فريق باعتباره علاقة بين ثلاثة أشياء: بنية معرفية تحدد ما يمكن تمثيله، حقل أفعال يحدد ما يمكن تغييره، ومحكمة تحقق تمنع تحويل الاحتمال اللغوي إلى قرار غير مختبر. إذا غابت المحكمة صار النظام مولداً للانطباعات. وإذا غابت الذاكرة صار النظام بلا تاريخ. وإذا غابت طبقة الحافز صار الوكيل تابعاً لأوامر سطحية لا يعرف كيف يولد هدفاً جديداً.

الاختراق المقترح. جوهر خطة بناء فريق هو نقل مركز الثقل من النموذج إلى الحلقة. الشركة الكبرى قد تمتلك نموذجاً أقوى، لكنها غالباً تدفع ثمناً في بطء التجريب، صعوبة إعادة بناء المعرفة، وتشتت فرق المنتج والبحث. أما المعمارية المقترحة فتجعل كل حلقة تشغيل تنتج أثراً قابلاً للقياس: هل تحسنت الخطة؟ هل انخفضت كلفة التفكير؟ هل زاد الاتساق السببي؟ هل تحولت الخبرة إلى قاعدة قابلة للنقل؟ هذه الأسئلة تجعل النظام يتطور ككائن بحثي لا كواجهة دردشة.

التطبيق العملي. في شركة تنافس المختبرات الكبرى، لا نبدأ ببناء نموذج عملاق، بل ببناء نظام مركب يحول كل تجربة إلى أصل قابل لإعادة الاستخدام. كل مهمة يجب أن تنتج: مساراً، حكماً، قاعدة، فشلاً موثقاً، وتعديلاً في المنهج. عندها يصبح المختبر الصغير أسرع من المؤسسة الضخمة لأنه يراكم خبرة قابلة للضغط والتوسيع.

الآثار الهندسية. لا يسمح هذا المفهوم ببنية عشوائية. يلزم مستودع traces، طبقة event sourcing، ذاكرة graph، مدقق causal، وسجل audit. كل قرار يجب أن يُخزن مع: المدخل، الخطة، البدائل التي رُفضت، سبب الرفض، المراجعة اللاحقة، وتأثيره على السياسة. بذلك يتحول التشغيل إلى تدريب، ويتحول التدريب إلى بحث، ويتحول البحث إلى منتج.

المؤشر التنافسي. نقيس خطة بناء فريق عبر ثلاثة مؤشرات: زمن الوصول إلى حل مقبول، نسبة الأخطاء المتكررة بعد التعلم، وقابلية نقل الحل من مجال إلى مجال. إذا تحسنت المؤشرات الثلاثة، فالمفهوم ليس تنظيراً؛ إنه آلية تنافسية.

صياغة رياضية مقترحة:

$$G_{consistency}=1-\frac{contradictions}{claims+evidence+1}$$

حيث تمثل الحالة s_t معرفة الوكيل الحالية، وتمثل الذاكرة M الخبرات المضغوطة، بينما تمثل الكلفة C ميزانية التفكير. الهدف ليس تعظيم الدقة فقط، بل تعظيم الدقة المشروطة بالكلفة والمخاطر وقابلية التفسير.

نواة تنفيذية مصغرة:

```
agent_bus.publish(intent)
critic = agents['critic'].review(intent)
if critic.risk < budget:
    executor.run(intent)
logger.commit(intent, critic)
```

ملاحظة هندسية: هذا الكود ليس منتجاً نهائياً، بل مخطط تحكم. يجب أن ينفذ داخل حاوية مع سجل أحداث، طبقة صلاحيات، مدقق أدوات، وذاكرة قابلة للإلغاء والتصحيح.

بروتوكول تدريب واختبار: اختبره على مهام طويلة لا تكفي فيها نافذة السياق: بحث، كود، قرارات متعددة المراحل، ثم قارن الذاكرة العادية بذاكرة الخبرة. في حالة خطة بناء فريق نضيف مقياساً خاصاً: هل يتحسن النظام بعد عشرين مهمة متتابعة دون إعادة ضبط يدوي؟ وهل يستطيع تفسير لماذا غيّر ذاكرته أو سياسته؟

خطة تحويل المفهوم إلى أصل تنافسي:
المرحلة الأولى: بناء نسخة مصغرة تعمل على عشر مهام فقط، مع تسجيل كل trace. المرحلة الثانية: تحويل أفضل traces إلى قواعد وسياسات. المرحلة الثالثة: إدخال مدقق خصومي يحاول إسقاط القرار قبل التنفيذ. المرحلة الرابعة: مقارنة النظام ب baseline بسيط لا يملك ذاكرة ولا تحققاً سببياً. لا يمر المفهوم إلى المنتج إلا إذا أظهر تحسناً في التعلم من الخطأ لا في الإجابة الأولى فقط.

نقطة الكسر. يفشل خطة بناء فريق إذا زادت كلفة الإدارة عن فائدة التعلم، أو إذا تحولت الذاكرة إلى مقبرة معلومات، أو إذا صار المدقق يقتل الإبداع بحجة السلامة. لذلك نحتاج ميزانية واضحة: جزء للتوليد، جزء للتحقق، جزء للتجريب، وجزء للنسيان. الذكاء ليس زيادة كل شيء؛ الذكاء هو توزيع الندرة.

فريق التنفيذ. يحتاج هذا المحور إلى أربعة أدوار: مهندس وكلاء، باحث تدريب، مهندس بيانات وسجلات، وناقد أمان. كل دور ينتج artifact أسبوعياً: كود، تجربة، تقرير فشل، وقرار تعديل. هكذا يتحول المشروع من فكرة كبيرة إلى مختبر يملك سرعة تكرار عالية.