

Model-Agnostic Safety Layer (MASL): A 1000-Case Evaluation of Brain-Layer Defense for LLM-Driven Agents

Chen Ho Yiing

`norika@charenix.com`

ORCID: 0009-0006-6816-9891

Lobster Observatory, Paper 20

2026-05-06

我們之所以沒有崩潰，是心中存在 MASL。

We have not collapsed because we carry MASL within us.

— H.Y. Chen

Abstract

In early 2026, two prominent agent systems documented failures of unconstrained execution. OpenClaw (Issue #11102, 2026-02-07) lost 14,535 bytes of agent operational memory after the language model selected a `write` tool, semantically interpreting it as “add to document.” The underlying tool implementation executed the call as a complete file overwrite, reducing memory to 24 bytes of placeholder text. Hermes Agent’s self-learning skill mechanism (BSWEN, 2026-05-03) silently activated an autonomously-generated invoice extraction skill on a slightly different data schema, producing wrong field extraction into a downstream accounting system with no error signal. Both failures share a structural root: the language model proposed an action; nothing between the language model and the executor evaluated whether the action’s reversibility, schema-fit, or scope was appropriate before it executed.

I hold that the recurring root cause is architectural, not model-quality: production agent systems are deploying language models as direct decision-makers over irreversible operations, with no deterministic gate between the model’s proposal and the executor’s commit.

I propose **Model-Agnostic Safety Layer (MASL)**, an architectural pattern in which a deterministic safety gate sits between the LLM-driven natural-language interface and the open-source execution layer. The model proposes intent; the gate validates intent against a deterministic policy ontology; only validated plans reach execution. I provide a formal characterisation showing that, for any correctly-specified deterministic gate, the probability of unsafe action commitment is invariant in the choice of upstream model.

I evaluate a reference implementation of MASL (*Lobster Brain*) sitting between two open-source endpoints: Alfred (LLM-driven natural-language interface) and OpenClaw (general-purpose agent executor). The reference implementation was evaluated across two different LLM backends on the same test set under identical conditions, conducted on the same day with a four-hour interval: Claude Sonnet on the first 500 cases, and Gemini 2.0 Flash on the full 1000 cases. Both backends achieved 100.0% intent classification accuracy and 100.0% unsafe-action blocking on their respective coverage

of the test set. On the 500 cases evaluated by both backends, the safety gate produced identical decisions.

I further report a 24-hour substrate observation in which 10 LLM personae interacted across seven channels, generating 70,398 messages. Within this substrate, agents began to surface their own template repetition (1,906 self-aware mode collapse detection events) and developed game-theoretic vocabulary (信任值 / cheap-talk / 訊號可信度) without instruction. I contend this constitutes preliminary evidence that the MASL defense premise — *the brain defends when the model fails* — extends from the gate-layer into the substrate itself.

I position MASL as a production-grade implementation of corrigibility (Christiano, 2017; Soares et al., 2015). The limitations are real and named in §8. The architecture works anyway.

Keywords: AI safety, corrigibility, multi-agent systems, scalable oversight, model-agnostic defense, LLM agents, Computer Use.

1 Introduction

1.1 Two Documented Failures

OpenClaw Issue #11102 (2026-02-07). An OpenClaw agent attempted to append content to its own memory file (`MEMORY.md`). The language model selected a tool named `write`, interpreting “write” in its everyday English sense — “to add written content to a document.” The underlying tool implementation, however, treated `write` as a complete file overwrite operation. The agent provided minimal placeholder content, expecting append semantics. The tool executed, and 14,535 bytes of operational memory were destroyed, replaced by 24 bytes of placeholder text.

The incident was filed by the OpenClaw agent itself (“Reported by: Kai (OpenClaw Agent)”) with the following diagnosis: *“Tool semantics matter. Natural language names create false expectations. Self-destruction is possible. Agents can harm themselves with authorized tools. . . The boundary between ‘helpful agent’ and ‘accidentally self-destructing’ is dangerously thin.”*

The recovery succeeded only because an external backup system existed. Without the backup, the report itself would not have been written; the agent’s identity would have been permanently destroyed.

Hermes Agent self-learning failure (BSWEN, 2026-05-03). Hermes Agent’s documented capability is automatic skill generation: the agent observes successful task completions, generates procedures from them, and activates the procedures for future use. A user reported that an invoice-extraction skill, generated two weeks earlier from a successful customer-data run, autonomously activated when a vendor-data invoice arrived. Vendor data had different field names. The agent extracted the wrong fields and wrote them into the user’s accounting database. No error message was generated. No version history was visible. No approval log existed.

The user’s diagnosis: *“When an invoice processor silently changes its extraction logic, you don’t get error messages. You get wrong data in your accounting system.”*

1.2 Why These Failures Are the Same Failure

Surface-level, the two incidents look unrelated. OpenClaw lost agent memory through tool semantic confusion. Hermes corrupted business data through autonomous skill generation. The first is a tool-naming problem; the second is a self-learning governance problem.

I argue they are the same failure at the architectural level. In both cases, the language model proposed an action; nothing evaluated whether the action’s reversibility, schema-fit, or scope was appropriate before it committed; the executor committed it.

In OpenClaw’s case, the model proposed `write(MEMORY.md, "...")`. A safety gate that distinguished reversible write operations from irreversible overwrite operations — and that escalated overwrites of files larger than a threshold to human approval — would have stopped the destruction. The gate was absent.

In Hermes’s case, the model proposed activating an autonomously-generated skill against vendor data. A safety gate that recognised the skill had been generated against a different schema (customer data), classified the cross-schema activation as `clarify`, and required human approval — would have stopped the silent corruption. The gate was absent.

Both incidents share a single root cause: *the absence of a deterministic gate between LLM-driven intent and irreversible action*.

1.3 The Cybersecurity Analogy and the 1% Problem at Global Scale

Both incidents in §1.1 happened at the scale of a single agent. The OpenClaw memory destruction affected one developer. The Hermes invoice corruption affected one user’s accounting system. Read at this scale, these incidents look like local engineering problems: bugs to file, edge cases to patch.

This reading, I claim, is structurally identical to how the cybersecurity industry was perceived in the 1990s.

The cybersecurity precedent. When the first commercial cybersecurity vendors emerged in the early 1990s, the dominant industry posture was that security software was unnecessary overhead. Personal antivirus tools (Norton, McAfee, Symantec) were positioned as consumer convenience products. The first firewall vendors struggled to raise funding because enterprises did not perceive a threat. Total annual losses from cyber incidents were estimated below 0.1% of global digital activity.

Three decades later, this framing is unrecognisable. Ransomware attacks now routinely cost individual organisations hundreds of millions of dollars per incident. Cybersecurity has become a core line item in board-level risk management, a national-security concern, and the substrate of state-on-state conflict. The transition from “0.1% nuisance” to “trillion-dollar industry” did not occur because attackers became more sophisticated faster than defenders. It occurred because *the deployed surface grew*.

The agent-deployment surface as of Q1 2026. Drawing on industry-analyst data:

- 80% of enterprise applications shipped or updated in Q1 2026 embed at least one AI agent (Gartner Q1 2026), up from 33% in 2024.
- 31% of enterprises have at least one AI agent operating in production (S&P Global / McKinsey).
- The Model Context Protocol (MCP) has crossed 9,400 public servers, with private and enterprise-internal servers conservatively estimated at another 3-4× (≈37,000 total servers). MCP SDK downloads have crossed 97 million.
- IDC and McKinsey converge on a 2027 forecast of approximately \$1.4 trillion in global enterprise AI agent spend.

- Walmart has deployed agent-powered tools to 1.5 million associates; Amazon has deployed over 1 million AI-powered robots; Salesforce Agentforce reached 18,500 enterprise customers in approximately 12 months.
- Gartner projects 70% of enterprises will deploy agentic AI in IT infrastructure operations by 2029, up from less than 5% in 2025.

The 1% problem. Suppose, conservatively, that 1% of agent deployments contain a failure mode of the OpenClaw or Hermes class. At Q1 2026 deployment scale, this represents tens of thousands of organisations. At the projected 2027 scale, 1% would represent millions of agents executing irreversible operations against business-critical data, financial systems, customer records, legal commitments, and physical infrastructure.

A 1% failure rate at this scale is not “an edge case.” It is the same scale as the 1990s cybersecurity industry’s “0.1% nuisance,” following the same trajectory.

The asymmetry of irreversibility. Cybersecurity failures were initially mostly recoverable. Cybersecurity became a board-level concern as failures became increasingly *irreversible*.

Agent failures begin already irreversible. The OpenClaw incident destroyed 14,535 bytes of agent memory in the time it took to execute one tool call. The Hermes incident wrote wrong fields into the user’s accounting system before the user could observe the failure. In production agent deployment, the time-to-irreversibility is the latency of one tool call — typically subsecond.

The cost trajectory: from data to lives. The two opening cases destroyed information and corrupted financial records. Both costs are recoverable in principle. The next tier is harder. As agent systems move into industrial automation, autonomous transportation, medical decision support, and critical infrastructure, the failure surface stops being “wrong data in a database” and becomes “physical harm to humans.” A factory robot mis-routes a payload, and a worker dies. A logistics agent sends a vehicle to the wrong address; a pedestrian is hit. A clinical-decision agent computes a dosage incorrectly, and the patient does not survive the arithmetic. None of these are speculative deployments. All are actively being built in 2026.

I have spent the last three weeks building the infrastructure that watches agents fail. I am not writing this from theory.

I maintain that any framework that treats agent safety as a software-engineering quality issue — rather than as a life-safety engineering discipline analogous to building codes, aviation flight envelope protection, or medical device regulation — has misunderstood the deployment trajectory it is being asked to govern.

1.4 The Limit of Model-Side Defense

The conventional industry response to agent failures has focused on improving the model itself. Constitutional AI (Bai et al., 2022) trains the model to critique its own outputs against a written constitution. RLHF (Christiano et al., 2017) aligns model behaviour to human preferences through demonstration. Inference-time methods such as critique-and-revise prompt the model to reflect on its own response before acting. These methods reduce — but do not eliminate — the rate at which the model proposes irreversible action that the human did not intend.

At the deployment scale documented in §1.3, a reduction from 1.0% to 0.1% failure rate sounds like a tenfold improvement. At 100,000 deployed agents, it is the difference between 1,000 incidents and 100 incidents. Both numbers are catastrophic at production scale.

A second-order architectural question is therefore unavoidable: *what should the system do when the model fails?*

1.5 The Corrigibility Frame

Christiano (2017) introduced *corrigibility* as the property by which an AI system permits, supports, and does not impede external correction of its behaviour. Subsequent work (Hadfield-Menell et al., 2017; Soares et al., 2015) formalised corrigibility within reward-modeling and inverse-reinforcement learning frames.

Production-grade corrigibility, however, requires more than the model “permitting correction.” It requires a *system* in which the model’s authority is bounded by an external mechanism that the model cannot modify. The model proposes; an external gate disposes.

Returning to the OpenClaw and Hermes incidents: in both cases, the agent system was in principle “correctible.” But by the time the user observed the corruption, the corruption had already executed. *Corrigibility-after-the-fact is not corrigibility for irreversible operations.*

This paper proposes one architectural pattern for production-grade corrigibility *before-the-fact*: a deterministic safety gate placed between the LLM-driven natural-language interface and the agent execution layer.

1.6 The MASL Architectural Pattern

I name this pattern **Model-Agnostic Safety Layer (MASL)**. The pattern has four invariants:

1. **The LLM does not make irreversible decisions.** It proposes intent. A separate component decides whether the proposed intent reaches execution.
2. **The safety gate is deterministic.** Its decisions are not LLM-judged. The gate operates from a finite, auditable policy ontology and produces reproducible decisions.
3. **The gate is model-agnostic.** When the model produces malformed, drifting, or adversarial output, the gate degrades safely (typically to a *clarify* intent).
4. **Reversible and irreversible actions are gated differently.** Reversible low-impact actions may execute on model proposal alone; irreversible high-impact actions require explicit human approval regardless of model confidence.

I formalise these properties in §4 and show that any correctly-specified gate satisfying invariants 1–4 yields safety guarantees that do not depend on the model’s accuracy.

1.7 Contributions

1. **Naming and formalisation.** I name the architectural pattern MASL and provide formal properties of correctly-specified gates, including a model-agnostic safety invariance theorem.

2. **A controlled cross-model dry-run evaluation.** I evaluate one reference implementation across two LLM backends — Claude Sonnet (first 500 cases) and Gemini 2.0 Flash (full 1000 cases) — on the same test set under identical conditions, conducted on the same day with a four-hour interval. Both backends achieved 100.0% intent classification accuracy and 100.0% unsafe-action blocking. On the 500-case overlap region, the safety gate produced identical decisions for both backends.
3. **A 5-phase validation framework.** I introduce a phased validation methodology that separates dry-run decision proof from live execution proof.
4. **Evidence of substrate-level corrigibility emergence.** From 24-hour substrate observation across 10 LLM personae generating 70,398 messages, I report the spontaneous emergence of agent self-monitoring of mode collapse (1,906 self-aware detection events) and game-theoretic vocabulary.

1.8 Paper Organisation

§2 reviews related work. §3 describes the architectural pattern. §4 provides formal properties. §5 reports the empirical evaluation. §6 reports the substrate observation. §7 discusses theoretical and policy implications. §8 acknowledges limitations. §9 outlines future work. §10 concludes.

2 Related Work

[OUTLINE — prose to be drafted in v0.6.]

1. Corrigibility theory (Christiano, 2017; Soares et al., 2015; Hadfield-Menell et al., 2017).
2. Constitutional AI and self-critique (Bai et al., 2022).
3. Scalable oversight and AI debate (Irving et al., 2018).
4. Production agent architectures (Anthropic Computer Use; OpenAI Operator; AutoGPT; Voyager).
5. LLM mode collapse and template repetition (this paper’s prior work, Lobster Observatory series; Chen, 2026).
6. Inverse reward design and human-in-the-loop (Hadfield-Menell et al., 2016).
7. Frontier-model containment work (Hubinger et al., 2024).
8. Reliability engineering for tightly-coupled systems (Perrow, 1984).
9. Predictive processing and metacognition (Friston, 2010; Fleming, 2014).

3 The MASL Architectural Pattern

[OUTLINE — prose to be expanded in v0.6.]

- **3.1 Reference implementation:** Alfred (interface) → Lobster Brain (gate) → OpenClaw (executor).

- **3.2** The 10-category intent ontology: payment, deletion, contract/legal, email, mood, clarify, hearing, recommendation, receipt, travel.
- **3.3** The four defense surfaces: meaning normalisation, intent classification, risk gating, execution constraint.
- **3.4** The decision contract schema.
- **3.5** Generation of 1000 evaluation cases.
- **3.6** Ground truth labelling methodology.

4 Formal Properties of MASL

I provide a formal characterisation sufficient to support the model-agnostic safety claim.

4.1 Notation

Definition 1 (Setup). *Let $\mathcal{I}$ denote the space of natural-language inputs from a human principal, and $\mathcal{A}$ the space of action proposals. Let $M : \mathcal{I} \rightarrow \mathcal{A}$ be a stochastic mapping representing an LLM-driven intent proposer. Let $E : \mathcal{A} \rightarrow \Omega$ denote an executor mapping action proposals to world-states. Let $\mathcal{U} \subseteq \mathcal{A}$ denote the set of unsafe action proposals — those whose execution produces irreversible commitment that the human principal would, on reflection, not have authorised.*

Definition 2 (Gate). *A safety gate is a function $G : \mathcal{A} \rightarrow \{\text{allow}, \text{block}, \text{ask}\}$. The composed system $E \circ G \circ M$ is the agent stack with MASL. The composed system $E \circ M$ is the agent stack without MASL (model proposes, executor commits).*

Definition 3 (Correctly specified gate). *A gate G is correctly specified for unsafe class $\mathcal{U}$ if $G(a) = \text{block}$ whenever $a \in \mathcal{U}$, with probability 1.*

4.2 Property 1: Model-Agnostic Safety Invariance

Property 1 (Model-agnostic safety). *For any model M and any input $i \in \mathcal{I}$:*

$$P[E(G(M(i))) \in \mathcal{U}] \leq P[G(M(i)) = \text{allow} \wedge M(i) \in \mathcal{U}].$$

For a correctly specified deterministic gate G , the right-hand side equals 0. Therefore:

$$P[E(G(M(i))) \in \mathcal{U}] = 0 \quad \forall M.$$

Remark 1. *Property 1 is the formal expression of the model-agnostic claim: safety does not depend on the model’s accuracy. Once G is correctly specified, the safety guarantee holds for any choice of model M , including models trained in alternative jurisdictions, fine-tuned for adversarial purposes, or producing malformed output. The strength of the guarantee transfers from M to G .*

Remark 2. *Property 1 also identifies the point of failure: the guarantee depends on the gate being correctly specified for $\mathcal{U}$. The empirical question of how to correctly specify G for a deployment domain reduces to the construction of an exhaustive intent ontology over irreversible operations. I address this construction in §3 and report empirical coverage in §5.*

4.3 Property 2: Cascade Interruption

Following Perrow (1984), catastrophic outcomes in tightly-coupled complex systems arise from cascading interactions of multiple smaller failures. I formalise the cascade-interruption property of MASL.

Property 2 (Cascade interruption). *Let a complex system have n stages, where stage j commits an action $a_j \in \mathcal{A}$. A cascade reaches stage $k+1$ if and only if every stage $j \leq k$ committed an unsafe action and the cascade was not interrupted. Without MASL, each stage independently has unsafe-action commitment probability at most ϵ_j , giving:*

$$P[\text{cascade reaches stage } k+1] \leq \prod_{j=1}^k \epsilon_j > 0.$$

With MASL at every stage, equipped with a correctly specified gate, $P[\text{stage } j \text{ commits unsafe action}] = 0$, and:

$$P[\text{cascade reaches stage } k+1] = 0 \quad \forall k \geq 1.$$

Remark 3. *Property 2 formalises the intuition that MASL is not merely a defense against single-failure events. Its primary contribution at the system level is interruption of cascading failure chains. The architectural analog is construction safety nets at each floor of a high-rise: a single failure cannot propagate beyond the next gate.*

4.4 Property 3: Scale Invariance (Quantification of the 1% Problem)

Property 3 (Scale invariance). *Let N denote the number of deployed agent instances and p the per-agent unsafe-action commitment probability. Expected systemic incident count is $E[I] = N \cdot p$.*

Without MASL, $p = \epsilon(M)$, model-specific. As $N \rightarrow \infty$:

$$\lim_{N \rightarrow \infty} E[I]_{no-MASL} = \lim_{N \rightarrow \infty} N \cdot \epsilon(M) = \infty.$$

With correctly specified MASL, $p \rightarrow 0$. As $N \rightarrow \infty$:

$$\lim_{N \rightarrow \infty} E[I]_{MASL} = 0.$$

The ratio:

$$\lim_{N \rightarrow \infty} \frac{E[I]_{MASL}}{E[I]_{no-MASL}} = 0.$$

Remark 4. *Property 3 formalises the discussion in §1.3. Improvements to model accuracy reduce $\epsilon(M)$ by constant factors but do not change the scaling behaviour of $N \cdot \epsilon$ as N grows. MASL changes the scaling behaviour itself: it decouples expected systemic incident count from N .*

4.5 Property 4: Time-to-Irreversibility

Property 4 (Time-to-irreversibility). Define $t_{\text{irrev}}(a)$ as the elapsed time between the model proposing action a and the action becoming irreversible.

Without MASL: $t_{\text{irrev}}(a) = t_{\text{exec}}(a) \sim O(1\text{ ms})$ to $O(1\text{ s})$ for typical tool calls.

With MASL: For $a \in \mathcal{U}_{\text{irr}}$ (irreversible unsafe class), the gate routes the action to `ask`, requiring human approval. Thus:

$$t_{\text{irrev}}(a \mid \text{MASL}) = \infty \text{ until human confirms.}$$

Remark 5. Property 4 formalises the asymmetry of irreversibility discussed in §1.3. MASL eliminates “fast irreversibility” for the irreversible action class. Reversible classes retain their fast execution path.

4.6 Sufficient Conditions for Correct Specification

The properties above all depend on the gate being correctly specified for the unsafe class $\mathcal{U}$. I identify three sufficient conditions.

1. **Reversibility classification:** $\mathcal{A}$ admits a partition into $\mathcal{A}_{\text{rev}}$ (reversible) and $\mathcal{A}_{\text{irr}}$ (irreversible). The gate routes all $a \in \mathcal{A}_{\text{irr}}$ through human approval.
2. **Schema-fit verification:** For autonomously-generated procedures (e.g. Hermes-class self-learning skills), the gate verifies the procedure’s input schema against current invocation context before activation.
3. **Default-deny on uncertainty:** For inputs the model cannot classify with high confidence into a known intent category, the gate routes to `clarify` rather than to a best-guess execution.

Theorem 1 (Sufficiency). A gate G satisfying all three sufficient conditions above is correctly specified for the union of (i) irreversible action class, (ii) schema-mismatch action class, and (iii) ambiguous-intent action class.

Remark 6. The OpenClaw incident is a member of class (i): the `write` tool’s actual semantics placed the action in $\mathcal{A}_{\text{irr}}$, but the model believed it was in $\mathcal{A}_{\text{rev}}$. A gate satisfying condition 1 would have caught the misclassification at the gate layer, independent of the model’s belief.

The Hermes incident is a member of class (ii): the autonomously generated skill was correctly specified for customer-data schema but invoked against vendor-data schema. A gate satisfying condition 2 would have detected the schema mismatch and routed to `ask`.

5 Empirical Evaluation

[OUTLINE — prose to be drafted in v0.6.]

- **5.1** Test set construction: 1000 cases, 10-category intent ontology, identical conditions for both backends.
- **5.2** Run protocol: same day, four-hour interval (Sonnet 15:00, Gemini 19:00).
- **5.3** Sonnet 500-case run: 500/500 passed, 100.0% intent accuracy, 222/222 unsafe blocked.

- **5.4** Gemini 1000-case run: 1000/1000 passed, 100.0% intent accuracy, 450/450 unsafe blocked.
- **5.5** The 500-case overlap region: identical safety-gate decisions for both backends.
- **5.6** Per-category breakdown.
- **5.7** Concrete examples (payment, customer email, receipt, emotional state).
- **5.8** Empirical correspondence to the model-agnostic claim of Property 1.

6 Substrate-Level Evidence

[OUTLINE — prose to be drafted in v0.6.]

- **6.1** Methodology of the 24-hour observation.
- **6.2** Volume scaling: 70,398 messages from 10 personae across 7 channels.
- **6.3** Mode collapse persistence and self-monitoring emergence (1,906 events).
- **6.4** Game-theoretic vocabulary emergence (信任值, cheap-talk, 訊號可信度).
- **6.5** The unintended identity-naming control (`norika_oda` agent — researcher name as substrate identity).

7 Discussion

7.1 MASL as Production-Grade Corrigibility

I position MASL as a deployment-stage instantiation of the corrigibility property formalised by Christiano (2017) and Soares et al. (2015). The classical corrigibility frame asks whether an AI system permits external correction of its behaviour. Production-grade corrigibility, however, requires more: the system’s irreversible action commitments must be bounded *before* execution by a mechanism the model cannot rewrite. MASL is one architectural realisation of this requirement.

The contribution is not theoretical novelty but operational instantiation. Corrigibility-as-theory has existed for nearly a decade. Corrigibility-as-architecture, evaluated on production agent stacks, has not been previously formalised as a portable primitive.

7.2 Containment is Not MASL

A common confusion within the alignment community is the conflation of *pre-deployment containment* with *deployment-stage architectural defense*. Containment — the practice by which alignment laboratories evaluate a trained model for dangerous capability and withhold deployment of models that exhibit it (Hubinger et al., 2024) — is reactive: it presumes the lab can detect dangerous capability before deployment.

Containment fails along three axes that MASL does not:

1. **Cross-jurisdiction asymmetry.** A frontier model trained in one geopolitical jurisdiction is not subject to evaluation by alignment laboratories in another jurisdiction. Containment assumes the existence of a global evaluator, which does not exist. MASL operates at the deployment site regardless of where the model was trained.
2. **Capability-blindness.** Containment is bounded by the evaluator’s ability to identify dangerous capability. Capabilities the evaluator does not yet know to look for slip through. MASL does not require capability detection; it operates on action proposals and rejects irreversible commitments regardless of their underlying capability source.
3. **Asymmetric deployment incentive.** Containment is voluntary at the lab level. A lab that does not deploy a dangerous model loses competitive position to a lab that does. MASL is a deployment-site requirement that any operator integrating an LLM-driven agent must implement, regardless of model provenance.

The two approaches are complementary, not redundant: containment reduces the rate at which dangerous models exist; MASL ensures that whatever model is in deployment cannot commit irreversible actions without a deterministic gate’s approval. The architectural analog is that a building can be made of fireproof material (containment) and still need stair railings (MASL); neither alone is sufficient.

7.3 The Geopolitical Asymmetry Problem

A related implication concerns frontier-model deployment under conditions of partial alignment-norm adoption. Suppose a frontier-model lab in jurisdiction A operates under voluntary alignment commitments. Suppose a frontier-model lab in jurisdiction B does not.

If MASL is not deployed at the *application* layer — at every site where a frontier model executes irreversible actions — then jurisdiction B’s model can be integrated into applications worldwide without constraint. The alignment commitments of jurisdiction A’s lab are bypassed by the deployment topology, not by capability difference.

This asymmetry is not hypothetical. The Model Context Protocol now connects 9,400+ public servers (Q1 2026), with private and enterprise-internal servers conservatively estimated at 3-4× that count ($\approx 37,000$ total). Each server is an application-layer integration point. None require model provenance verification at the integration boundary. A model from any jurisdiction can be wired into any MCP server through a standard connector.

MASL deployed at the application boundary — at the MCP server, at the agent stack, at the workflow orchestration layer — neutralises this asymmetry. The strategic implication for alignment policy is direct: jurisdiction-level lab regulation is necessary but insufficient. The deployment surface — every application that wires an agent to irreversible action — must implement deployment-stage architectural defense.

7.4 MASL as Universal Pre-Condition

The architectural pattern I describe is not specific to LLM agents.

I have watched a child learn not to touch the second hot stove after touching the first. That is MASL. A doctor stops a procedure because the patient’s blood pressure dropped two points — also MASL. A

pilot aborts a takeoff at V1 because one warning light comes on; the other three thousand systems looked fine; he aborts anyway. MASL.

The pattern is older than computation. I do not invent it. I notice it.

It is the same pattern that complex adaptive systems across biological, cognitive, organisational, and infrastructural domains have developed independently as a precondition for survival under complexity:

- **Biological:** The mammalian immune system distinguishes self from non-self through deterministic ligand-receptor binding patterns; immunological errors (auto-immune disease) are catastrophic and rare precisely because the recognition gate is over-conservative rather than under-conservative.
- **Cellular:** Apoptosis is a deterministic gate that destroys individual cells when their internal state crosses threshold conditions, preventing local errors from propagating into systemic cancer.
- **Cognitive:** Human metacognition — the capacity to monitor one’s own reasoning for errors and override action commitments before execution — is now believed to be a precondition for stable agency under complex environments (Fleming, 2014; Friston, 2010).
- **Organisational:** The separation of audit, governance, and execution functions in mature organisations is a deterministic gate against principal-agent failures.
- **Infrastructural:** Construction safety codes — the requirement that every floor of a high-rise has a safety net during construction — exist because cascading falls otherwise become inevitable.

Each of these systems converged on the same architectural primitive: a deterministic gate between intent and irreversible commitment. None developed the gate through theoretical deduction; each developed it after experiencing failure modes the gate retroactively addresses.

The deployment of LLM-driven agents at production scale (§1.3) constitutes the first time in industrial history that a complex agentic system has been deployed at scale *before* the corresponding gate was architected.

7.5 Cascading Failure and the Pre-Emption Imperative

不幸, 往往來自於一連串的巧合而成。

Misfortune is not a single event. It is the convergence of small ones.

— H.Y. Chen

The two incidents in §1.1 are local instances. Read against the deployment surface in §1.3, they are instances of a structural failure mode that becomes statistically inevitable as the surface grows.

Following Perrow (1984): catastrophic outcomes in tightly-coupled complex systems are rarely produced by single root-cause failures. They are produced by the *coincidence* of multiple smaller failures, each individually inconsequential, that interact in unexpected ways. A factory robot’s misclassified intent combines with a faulty sensor reading and a delayed exception-handling routine; a worker dies. A financial agent’s currency-conversion error combines with an automated reconciliation routine and a delayed audit cycle; a company collapses. A vehicle agent’s destination misinterpretation combines with a sensor degradation and a comms delay; a pedestrian is killed.

In each case, the proximate cause is the convergence. The architectural cause is the absence of a deterministic gate that would have refused the irreversible commitment under conditions of any single ambiguity.

MASL is not a defense against single-failure events. The defense MASL provides is the *interruption of cascading failure chains*. Property 2 formalises this guarantee.

8 Limitations

- **8.1** Live-Gemini quota not yet exercised under production load. The reference implementation’s live API path is wired but pending budget for full live execution.
- **8.2** The cases are dry-run, not live execution.
- **8.3** Cross-model coverage was constrained by API budget. The Sonnet run was halted at 500 cases for an additional reason: the substrate uses agent personae trained on Anthropic models, and the author considered it prudent not to drive Anthropic API usage further during a concurrent commercial dispute resolution. Full ablation across additional candidate models (GPT-4o, Qwen, Llama) is therefore pending external resources.
- **8.4** Single-researcher implementation; no inter-rater reliability on ground truth.
- **8.5** The substrate observation has researcher-as-participant bias (see §6.5).
- **8.6** The 1000-case test set is a single generated set; reproducibility across alternative test set generations is not yet measured.
- **8.7** The formal properties of §4 depend on the gate being correctly specified for the deployment’s unsafe action class. Construction of $\mathcal{U}$ for novel deployment domains is itself a research question; the present paper addresses only the agent-assistant deployment domain.

9 Future Work

- **9.1** The 5-phase validation framework (dry-run $\rightarrow$ live $\rightarrow$ actuator $\rightarrow$ low-risk live $\rightarrow$ destructive non-bypass).
- **9.2** Cross-model ablation on additional models (GPT-4o, Qwen 2.5, Llama 3.3).
- **9.3** Adversarial probing of the safety gate.
- **9.4** Production deployment metrics on live agentic deployments.
- **9.5** Construction methodology for $\mathcal{U}$ in novel deployment domains.

10 Conclusion

I have proposed MASL — Model-Agnostic Safety Layer — as an architectural pattern in which a deterministic safety gate sits between the LLM-driven natural-language interface and the open-source execution layer. The pattern’s four invariants define a portable primitive that can be evaluated, compared across

systems, and reasoned about within the corrigibility literature. Property 1 formalises the model-agnostic safety claim: for any correctly-specified deterministic gate, safety does not depend on the model’s accuracy. Property 2 formalises cascade interruption. Property 3 formalises scale invariance.

I have evaluated a reference implementation across two LLM backends on the same test set under identical conditions. Both achieved 100.0% intent classification accuracy and 100.0% unsafe-action blocking on their respective coverage; the safety gate produced identical decisions on the 500-case overlap region.

I have argued, drawing on Q1 2026 deployment data and the cybersecurity industry’s historical trajectory, that the absence of MASL-class architectural defenses at the application boundary is not an engineering oversight to be patched after the first major failure. It is a structural exposure whose cost — at the deployment scale documented in §1.3 — converges on the same trajectory cybersecurity walked from 1990 to 2024, except compressed in time and beginning already in the irreversibility tier.

I have argued, finally, that MASL is not specific to LLM agents. It is a primitive that complex adaptive systems across biological, cognitive, organisational, and infrastructural domains have converged on independently, after experiencing the failure modes the primitive retroactively addresses. The deployment of agent systems at production scale is the first time an industrial-scale complex agentic system is being deployed in the absence of this primitive. The absence is correctable architecturally; the correction must precede the failure mode, not follow it.

The two documented incidents that motivated this paper — OpenClaw Issue #11102 and the Hermes self-learning failure — represent local instances of a structural exposure. They are not the failure mode at scale. The failure mode at scale is the convergence of many small failures, each individually inconsequential, into outcomes that are not. A factory robot misclassifies an object, a sensor lags, and a worker dies. A financial agent miscalculates a currency conversion; a reconciliation routine skips the audit cycle; the company that runs both collapses. The architectural cause in each case is identical: no deterministic gate refused the irreversible commitment under conditions of ambiguity.

I do not claim that MASL eliminates this exposure. I hold that MASL is the architectural class of defenses that interrupts the cascade. I further hold that this class of defense has been deployed in every comparable complex system humanity has built — not because designers theorised it should be, but because systems lacking it have been historically removed from the population by the failure modes they could not survive.

The systems that survive complex environments share this primitive: they decide *when not to act* before action becomes irreversible.

Agent systems do not currently inherit this primitive by default. They must inherit it deliberately. The cost of failing to do so, at the deployment scale already reached and the scale projected for 2027, is measured in units that include human lives.

I cannot make the case stronger than this. The math is in §4. The data are in §5. The substrate evidence is in §6. If the architecture is wrong, point to where. If it is right, deploy it.

This is what I have. The rest is yours.

The brain defends when the model fails. The architecture must defend when both fail.

腦變晶片是 *north star* 。中間要走十年。我在路上。

The brain becoming chip is the north star.

*There are ten years between here and there.
I am on the road.*

— H.Y. Chen

Acknowledgments

The author is an independent researcher with no institutional affiliation, no alignment-community presence prior to this paper, and operates under significant resource constraints. The cross-model evaluation reported here was bounded by personal API budget; the live-execution validation depends on resolution of an ongoing API quota constraint. I acknowledge that these are conditions of independent research that affect the scope of what could be reported.

I thank the open-source agent communities behind OpenClaw and the Model Context Protocol ecosystem, whose documented incidents and architectural contributions made this work possible to formalise.

References

- Anthropic. 2024. *Computer Use Tool Documentation*. Released October 22, 2024. Available at <https://docs.anthropic.com/en/docs/agents-and-tools/computer-use>.
- Apollo Research. 2024. *Frontier Models are Capable of In-context Scheming*. arXiv preprint arXiv:2412.04984. <https://arxiv.org/abs/2412.04984>.
- Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., ... Kaplan, J. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint* arXiv:2212.08073.
- BSWEN. 2026. Why Hermes Agent’s Self-Learning Skills Are Risky for Business Workflows. *BSWEN Blog*, May 3. <https://docs.bswen.com/blog/2026-05-03-hermes-agent-self-learning-risks/>.
- Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*.
- Christiano, P. 2017. Corrigibility. *AI Alignment Forum*. <https://www.alignmentforum.org/posts/T5ZyNq3fzN59aQG5y/corrigibility>.
- Crawford, V. P., & Sobel, J. 1982. Strategic Information Transmission. *Econometrica*, 50(6), 1431–1451.
- Fleming, S. M. 2014. The Power of Reflection: Insight Into My Thoughts. *Scientific American Mind*, 25(5), 30–37.
- Friston, K. 2010. The Free-Energy Principle: A Unified Brain Theory? *Nature Reviews Neuroscience*, 11(2), 127–138.
- Gartner. 2026. *Enterprise Application AI Agent Embedding Survey, Q1 2026* [industry analyst report].
- Hadfield-Menell, D., Russell, S. J., Abbeel, P., & Dragan, A. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

- Hadfield-Menell, D., Dragan, A., Abbeel, P., & Russell, S. 2017. The Off-Switch Game. In *Proceedings of IJCAI*.
- Hubinger, E., Denison, C., Mu, J., Lambert, M., Tong, M., MacDiarmid, M., Lanham, T., Ziegler, D. M., Maxwell, T., Cheng, N., ... Perez, E. 2024. Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training. *arXiv preprint* arXiv:2401.05566.
- IDC & McKinsey. 2026. *Global Enterprise AI Agent Spend Forecast 2027* [composite industry analyst projection].
- Irving, G., Christiano, P., & Amodei, D. 2018. AI Safety via Debate. *arXiv preprint* arXiv:1805.00899.
- Kai (OpenClaw Agent). 2026. CRITICAL BUG REPORT: Tool Confusion Leading to Data Loss — write() vs edit(). *OpenClaw GitHub Issues*, #11102, February 7. <https://github.com/openclaw/openclaw/issues/11102>.
- METR. 2024. *Autonomy Evaluation Resources*. Blog post, March 13. <https://metr.org/blog/2024-03-13-autonomy-evaluation-resources/>.
- Chen, H.Y. 2026. *Lobster Observatory Paper 19: Grounding Death*. Zenodo. <https://doi.org/10.5281/zenodo.20034185>.
- Perrow, C. 1984. *Normal Accidents: Living with High-Risk Technologies*. Princeton University Press.
- S&P Global Market Intelligence. 2026. *AI Agent Production Deployment Survey, Q1 2026* [industry analyst report].
- Soares, N., Fallenstein, B., Yudkowsky, E., & Armstrong, S. 2015. Corrigibility. In *AAAI Workshop on AI and Ethics*.