Verified AI Actions:
Closing the Pre-Action Legitimacy Gap

Thomas Dionysopoulos, CFA

Abstract

When an AI system selects a tool, its selection is an assertion. Every production tool-calling protocol—MCP, OpenAI function calling, LangChain—treats this assertion as sufficient for execution. No evidence is required. No verification occurs. The tool runs.

We identify a *verification gap* between what AI systems assert and what they can justify. This gap has been independently named from three directions: the *assertion gap* in tool selection [1], the *pre-action legitimacy gap* in regulatory compliance [6], and the requirement that "generation is not permission" in formal agent safety [3].

We map the emerging landscape of AI action verification into three layers: post-hoc audit (signed logs), policy gates (pre-execution permission checks), and semantic verification (pre-execution justification checks). Recent work implements integrity verification via content-addressed tool manifests [21] and policy verification via Z3-backed formal proofs [5] and pre-execution firewalls [16]. No deployed system implements the third layer: verifying that a selection is *semantically correct for the user's intent*, not merely authentic or permitted.

We present an implemented and empirically evaluated runtime semantic-verification system [1] with a verified-replay companion [2]. The system uses content-addressed *behavioral* identity (SHA-256 over semantic, algebraic, and implementation properties—excluding descriptions) and a runtime grounding wall that blocks unverified selections at execution time. In 180 trials, semantic verification reduced uncaught wrong-tool selection from 23.3% to 10.0%; the same behavioral identity hash eliminates false hits in compositional caching (0 vs. 97 in 1,000 pipelines). To our knowledge, this is the first implemented and empirically evaluated system for runtime semantic verification of AI tool selection. We define the requirements for a cross-framework verification protocol and outline the open problems that constitute this research program.

1 The Verification Gap

Consider the lifecycle of an AI tool call in any production system:

1. The user states a request in natural language.
2. The AI system selects a tool and generates arguments.
3. The tool executes.

Step 2 is an assertion: "this tool is appropriate for this request." In every major tool-calling protocol—Anthropic's Model Context Protocol [9], OpenAI function calling [10], LangChain [11], AutoGen [12]—this assertion flows directly to step 3. Schema validation may check that arguments are well-formed. Safety guardrails may check that the call is not dangerous. But no mechanism checks that the selection is *justified by the user's request*.

The tool may be structurally valid—it exists, it type-checks—but the user's request does not warrant it. A momentum strategy produces a mean-reversion signal. A Sharpe ratio metric is swapped for maximum drawdown. The output is correct in form and wrong in substance. Schema validation cannot detect this: both tools are valid enum values.

This gap between valid selection and justified selection has been identified independently from three directions in 2026:

- **Tool selection:** The *assertion gap*—the distance between a tool selection that is structurally valid and one that carries evidence linking it to the user's request [1].

- **Regulatory compliance:** The *pre-action legitimacy gap*—the distance between what regulators require (per-action accountability, EU AI Act Article 86, MiFID II order trails) and what exists (post-hoc logs) [6].
- **Formal safety:** The principle that "generation is not permission"—an AI system's proposal of an action does not constitute authorization to execute it [3].

Three communities, arriving independently at the same structural observation: AI actions are asserted, not verified.

2 Three Layers of Action Verification

The emerging landscape of AI action verification separates into three layers, distinguished by *what* is verified and *when*.

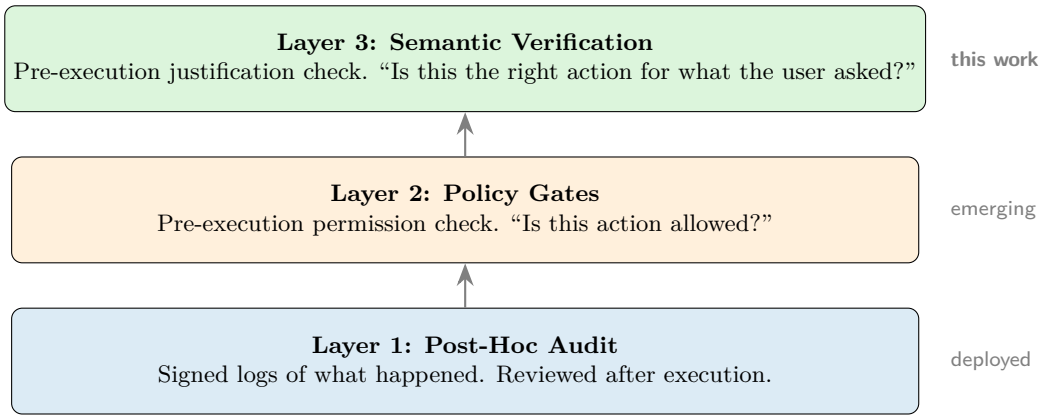


Figure 1: Three layers of AI action verification. Production systems implement Layers 1–2. This work implements Layer 3.

Layer 1: Post-hoc audit. Signed, tamper-evident records of executed actions. Examples: Auditable Agents [14] (Ed25519-signed, SHA-256 hash-chained records), Dapr 1.18 verifiable execution (SPIFFE-backed cryptographic receipts), Agent Receipts [15] (W3C Verifiable Credentials for agent behavior), and standard experiment tracking (MLflow [18], W&B [19]). Regulatory frameworks (ISO 42001 A.6.2.8, EU AI Act Article 12) require this layer. It answers: "what happened?" It does not prevent anything.

Layer 2: Policy and integrity gates. Pre-execution checks that verify an action is *permitted* or *authentic* before it runs.

Policy verification. ePCA [5] translates actions to first-order logic and verifies satisfiability against security axioms via Z3 (0.44 ms per decision, 100% attack detection; explicitly cites Necula [7]). AEGIS [16] interposes a pre-execution firewall with 22 risk-detection patterns (48/48 attacks blocked, 1.2% FPR, 8.3 ms median latency). Open Agent Passport [13] enforces Ed25519-signed per-action authorization (0% bypass in 4,437 decisions). NeMo Guardrails [17] enforces policy via Colang DSL execution rails.

Integrity verification. Zhou [21] implements content-addressed tool identity via SHA-256 manifest hashing with runtime hash verification (97 μs, zero false positives across 12 attack scenarios). This verifies that a tool has not been tampered with—the tool *is what it claims to be*.

All Layer-2 systems answer: "is this action allowed and authentic?" None answers: "is this the *right* action for what the user asked?" A wrong-but-permitted, authentic tool passes every Layer-2 check.

Layer 3: Semantic verification. Pre-execution checks that verify an action is *justified by the user's context*. This requires: (a) evidence linking the selection to the user's terms, (b) a verification function that checks this evidence, and (c) a wall property—no unverified action reaches execution. The Columbia PCE

architecture [3] formalizes the general principle that generation is not permission. Barrett et al. [4] describe the pattern: "AI produces artifact + certificate, checkable independently."

The distinction from Layer 2 is precise. Policy gates check *permission* (is this action within bounds?). Integrity gates check *authenticity* (is this tool what it claims to be?). Semantic verification checks *justification* (does this action have evidence linking it to what the user asked?). Content-addressed *manifest* identity [21] hashes over what a tool *is* (name, schema, version). Content-addressed *behavioral* identity [1] hashes over what a tool *does* (semantic, algebraic, and implementation properties—excluding descriptions). A tamper-free tool that does the wrong thing passes integrity verification but fails semantic verification.

Prior to this work, no implemented system provided Layer 3. Formal architectures exist [3, 4]. Policy and integrity implementations exist [5, 16, 21]. The semantic verification layer—where each action carries evidence of its justification, and that evidence is checked before execution—was unimplemented.

3 An Implemented Semantic-Verification System

We present a system that implements Layer-3 semantic verification with runtime enforcement, evaluated across 11 experiments in quantitative finance using Anthropic Claude models.

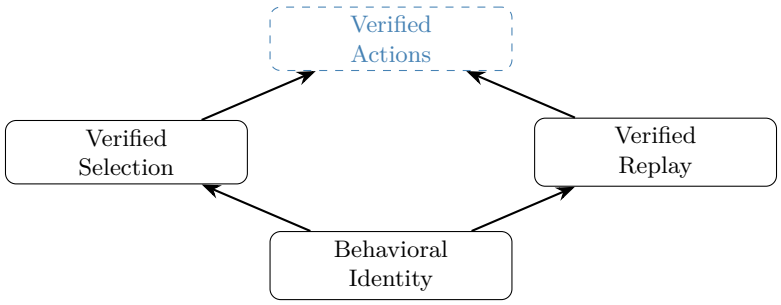
Verified tool selection [1]. Each tool in the registry has a content-addressed behavioral identity: a SHA-256 hash computed over its semantic class, algebraic properties, and implementation parameters—excluding natural-language descriptions. When an AI system selects a tool, a three-stage verification pipeline (discover → ground → dry-run) generates evidence linking the selection to the user's natural-language terms. A runtime grounding wall enforces the wall property: the execution engine checks that the tool's behavioral hash was verified by the dry-run stage, and blocks execution if it was not. This is not advisory—unverified selections produce a `GROUNDING_WALL` error and do not execute. The wall is tested by 9 property tests and is enforced across the full test suite (1,600 tests, 0 failures).

Result: On 30 prompts across 5 categories (180 trials, Claude Sonnet, $T = 0.7$), an assertion-only pipeline (schema validation, no verification) executes unwarranted tools at 23.3%. The verified pipeline reduces this to 10.0% (Fisher exact $p = 0.027$), eliminating unwarranted executions entirely on undiscoverable prompts (100%→0%). Verification overhead: under 14 ms per gate operation.

Verified replay [2]. A computation identity system where each capability is hashed over behavioral properties (semantic, algebraic, implementation), with discovery metadata excluded by construction. This identity hash is the foundation for an 8-layer execution hash that enables deterministic replay and fault localization.

Result: Safe compositional caching requires the cache key to induce a congruence on the pipeline space. A cache keyed on content hashes (DataHash) violates this—97 false hits in a 1,000-pipeline experiment. A cache keyed on identity hashes (incorporating the behavioral capability hash) induces a congruence—zero false hits.

The connection. These are not independent results. The behavioral identity hash that makes verification evidence stable (first system) is the same hash that makes replay safe (second system). Verified selection and verified replay share the same identity foundation:



Comparison with prior systems. Table 1 summarizes the verification landscape. The key column is the last: semantic verification—checking that the selected tool is *correct for the user's intent*, not merely authentic or permitted.

Table 1: Verification capabilities of AI tool-use systems. *Integrity*: tool has not been tampered with. *Policy*: tool is permitted by rules. *Semantic*: tool is correct for user intent.

System	Integrity	Policy	Semantic	Enforcement	Evidence
Zhou [21]	✓	—	—	✓	✓
AEGIS [16]	audit	✓	—	✓	✓
ePCA [5]	—	✓	—	✓	✓
OAP [13]	partial	✓	—	✓	✓
NeMo [17]	—	✓	—	✓	partial
Columbia PCE [3]	—	formal	—	—	—
Barrett et al. [4]	—	—	formal	—	—
MCP [9]	—	—	—	—	—
OpenAI FC [10]	—	—	—	—	—
This work	✓	✓	✓	✓	✓

Caveats. The system is evaluated in a single domain (quantitative finance, 236 capabilities), with a single model family (Anthropic Claude), and evaluated by the system authors. All experiments are reproducible: scripts, data, and expected results are committed to the repository. Enforcement is opt-in (`--require-grounding`). These are limitations of the current evidence, not of the architecture.

4 Requirements for a Verification Protocol

For semantic verification to become a protocol rather than a property of one system, four requirements must be met.

R1: Certificate format. Each tool call must carry a machine-readable certificate containing: the action identifier, the capability reference (name, behavioral hash, registry version), the evidence (match mode, confidence, discovery trace linking the selection to user terms), and a context hash binding the certificate to the user's request.

R2: Independent verifier. A deterministic verification function that accepts a certificate and a registry snapshot, and returns ACCEPT or REJECT. The verifier must be independent of the selection mechanism—it does not know or trust how the selection was made; it only checks whether the evidence is valid.

R3: Wall property. The architecture must enforce: for all actions a, execute(a) requires verify(cert(a)) = ACCEPT. This must be architectural (enforced by the runtime), not conventional (enforced by documentation).

R4: Framework independence. The certificate format and verifier interface must be independent of the underlying tool-calling protocol. The same certificate should be verifiable whether the tool call originated from MCP, OpenAI function calling, LangChain, or any other framework. This is the requirement that separates a protocol from a system.

The system presented in Section 3 satisfies R1–R3 within a single framework. No system satisfies R4. Cross-framework portability is the open engineering problem that separates the current state (one implemented semantic-verification system) from the target state (a verification protocol).

5 Open Problems

Cross-framework adapters. The immediate engineering challenge: wrap MCP tool calls, OpenAI function calls, and direct API calls with a common certificate format. The verification function must be protocol-agnostic. This is the experiment that would demonstrate the pattern is an architecture, not an implementation detail.

Third-party attestation. The current evidence model is self-generated: the system that selects the tool also generates the evidence. The structural analog to SSL/TLS breaks here—SSL certificates are third-party attested by certificate authorities. Whether AI action certificates require third-party attestation, and what entity would serve as the authority, is an open design question. One possibility: the capability registry itself, if operated independently of the AI system, serves as the trust anchor (analogous to a CA). Another: a standards body maintains a public registry of behavioral hashes (analogous to a trust store).

Formal verification. The wall property is currently enforced by implementation. Formally proving it—in the style of Necula's PCC [7] or ePCA's Z3-based verification [5]—would strengthen the guarantee. Justification Logic [8] ($t{:}\phi$, "t is evidence for ϕ") provides a candidate formalism but has not been applied to AI agent actions.

Semantic verification beyond tool selection. The current evidence addresses tool *selection* (is this the right tool?). Extending to tool *parameterization* (are these the right arguments?) and tool *composition* (is this the right sequence of tools?) requires richer evidence structures. The 8-layer execution hash provides partial coverage of composition; full compositional verification is open.

Regulatory alignment. The pre-action legitimacy gap [6] identifies the regulatory demand. Machine-readable compliance evidence formats (OSCAL [20]) are emerging but not yet connected to per-action verification. Bridging Layer 3 verification to regulatory reporting standards is a concrete integration problem.

Scaling. The existing evidence comes from a registry of 236 capabilities with 4 strategy families and 9 metrics. Whether semantic verification scales to registries with thousands of capabilities—where discovery ambiguity increases and the verification function faces harder matching problems—is empirically untested.

6 Conclusion

The verification gap in AI tool use is real, independently identified, and partially addressed—but only at the policy layer. Integrity verification confirms tools are authentic [21]. Policy verification confirms tools are permitted [5, 16, 13]. No prior implemented system confirms tools are *correct for the user's intent*.

This work provides the first implemented and empirically evaluated system for runtime semantic verification of AI tool selection. The key mechanism is content-addressed *behavioral* identity—hashing over what a tool does, not what it is—combined with a runtime grounding wall that blocks unverified selections at execution time. The system reduces uncaught semantic errors from 23.3% to 10.0%, and the same identity model eliminates false hits in compositional caching.

What remains is the protocol: a cross-framework certificate format, an independent verifier interface, and a second implementation demonstrating that the pattern generalizes beyond one system, one domain, and one model family.

The research program is: make Layer 3 a standard.

References

[1] T. Dionysopoulos. The grounding gate: Verified tool selection for AI-driven research. 2026.

[2] T. Dionysopoulos. Canonical execution semantics for stochastic program generators. 2026.

[3] Y. Liu, S. Wang, and A. Capponi. No certificate, no execution: A proposal-certification-execution architecture for AI agents. *arXiv:2605.24462*, May 2026.

[4] C. Barrett, T. A. Henzinger, and S. A. Seshia. Certificates in AI: Learn but verify. *Communications of the ACM*, January 2026.

[5] Z. Wu et al. Provably secure agent guardrail via evidence-based policy compliance. *arXiv:2605.29251*, May 2026.

[6] O. Lavi. The pre-action legitimacy gap in autonomous AI systems. *arXiv:2604.24153*, April 2026.

[7] G. C. Necula. Proof-carrying code. In *POPL*, 1997.

[8] S. N. Artemov. The logic of justification. *Review of Symbolic Logic*, 1(4):477–513, 2008.

[9] Anthropic. Model Context Protocol specification, 2024.

[10] OpenAI. Function calling and structured outputs, 2024.

[11] H. Chase. LangChain, 2023.

[12] Q. Wu et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv:2308.08155*, 2023.

[13] Open Agent Passport. Per-action authorization with cryptographic signing for AI agents. *arXiv:2603.20953*, March 2026.

[14] USC FORTIS Lab. Auditable agents: Cryptographically signed action records. *arXiv:2604.05485*, April 2026.

[15] Obsigna. Agent receipts: W3C Verifiable Credentials for agent behavior. 2026.

[16] AEGIS. Runtime guardrails with wall property for LLM tool calls. 2026.

[17] NVIDIA. NeMo Guardrails: Programmable rails for LLM applications, 2024.

[18] M. Zaharia et al. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Engineering Bulletin*, 41(4), 2018.

[19] L. Biewald. Experiment tracking with Weights and Biases, 2020.

[20] D. Cilla Ugarte et al. Machine-readable compliance evidence for AI systems via OSCAL. *arXiv:2604.13767*, April 2026.

[21] J. Zhou. Content-addressed tool identity via skills manifest hashing. *arXiv:2603.14332*, March 2026.