

Agentic Execution Assurance: Observation-Bounded Certification, Runtime Enforcement, and Tamper-Evident Evidence

Stamatis-Christos Saridakis

Independent Researcher

contact omitted for review draft

October 2026 – Revision 2.1

Manuscript DOI: [10.5281/zenodo.23119087](https://doi.org/10.5281/zenodo.23119087)

Reproducibility artifact: [10.5281/zenodo.23118665](https://doi.org/10.5281/zenodo.23118665)

Abstract

AI-agent evaluation increasingly measures consistency, robustness, predictability, safety, and task success across repeated trials. Those empirical questions are essential, but they are distinct from a systems-assurance question: *given a property, an observation boundary, and an enforcement boundary, what can an external party certify, what can a runtime gate prevent before side effects occur, and what evidence can support an independent audit?* We develop a bounded execution-assurance framework that connects specification, observability, enforcement, and evidence integrity for tool-using agents. Building on established ideas from rough-set approximations, three-valued runtime monitoring, partial-observability monitoring, and safety enforcement, we specialize observation-induced equivalence classes to agent execution traces and state an exact information-theoretic criterion: a property admits a sound-and-complete observation-only Boolean decision function on realizable observations exactly when it is constant on observation classes. The result is set-theoretic and does not by itself imply a computable verifier. We define conservative CERTIFIED/VIOLATED/UNKNOWNaudit semantics, with OUTOfMODEL reserved for malformed or unrealizable observations, and prove monotonicity under observation refinement. We then give a conditional prefix-safety theorem over the committed-execution projection under complete mediation, pre-effect checking, rejection stuttering, and state-consistent admission. We replace a universal scalar reliability score with a support-aware assurance profile and specify a draft Agentic Reliability Evidence Profile (AREF) that can be carried by OpenTelemetry while adding claim-dependent policy, commit, ordering, and integrity semantics. Finally, we audit AEGISRUN v1.0.1 as a concrete reference implementation. Its gateway and evidence stack instantiate substantial parts of the architecture, but strict guarantees remain bounded by deployment mediation, signer trust anchoring, budget-admission atomicity, post-effect egress accounting, approval/resume, and crash consistency. Controlled experiments over 2,254 matched traces illustrate the observation-class characterization on a disclosed finite universe; a paired 4,000-proposal simulation validates metric bookkeeping rather than detector effectiveness; nine post-root corruption/forgery fixtures are detected while an authorized-recorder omission control verifies; and a separate Python implementation reproduces the pinned Go golden vectors. The companion artifact was rerun for this revision and its invariant checks passed. The result is a scoped integration of established verification principles with an explicit evidence contract and a pinned agent gateway, intended to make assurance claims auditable without overstating what telemetry, signatures, or controlled simulations establish.

Reproducibility artifact. The exact Revision 2.1 archival artifact is publicly archived at DOI [10.5281/zenodo.23118665](https://doi.org/10.5281/zenodo.23118665).

1 Introduction

Tool-using AI agents act over time. They issue API requests, update files and databases, communicate with users and services, consume budgets, delegate subtasks, and accumulate state. Consequently, a final answer is only a projection of an execution. Two runs can return the same output while differing in whether a forbidden tool was attempted, a policy was bypassed, a rejected operation produced a side effect, or the audit record was modified after the fact.

The empirical reliability literature has now made the complementary point that task accuracy is too narrow. Rabanser et al. decompose agent reliability into consistency, robustness, predictability, and safety and evaluate repeated executions rather than a single aggregate score [17]; HAL standardizes large-scale agent evaluation across models, scaffolds, and benchmarks [8]; and log-analysis work shows that final pass/fail outcomes can conceal evaluation artifacts and unsafe actions [9]. We adopt that terminology rather than competing with it.

This paper addresses a different layer. We use *execution assurance* for the following question:

Given an execution property, an observation boundary, and an enforcement boundary, what can be positively certified, what can be refuted, what must remain unknown, and what violations can be prevented before irreversible effect?

The thesis is that agentic execution assurance requires four linked surfaces:

Specification + Observability + Enforcement + Evidence integrity.

A property without an observation contract may be unprovable after the fact. Observation without an enforcement boundary may detect but not prevent. Enforcement without evidence may be difficult to audit independently. Evidence without integrity may be altered after execution.

Contributions. This revision makes six scoped contributions.

1. It separates *behavioral/empirical reliability* from *execution assurance* and gives a claim-oriented vocabulary for certification, enforcement, auditability, integrity, and support.
2. It specializes established equivalence-class and partial-observability ideas to agent execution traces, states the exact set-theoretic Boolean-certifiability criterion on realizable observations, and makes uncertainty and out-of-model inputs explicit rather than conflating them with failure.
3. It states a typed conditional runtime-enforcement theorem over committed prefixes, clarifying the mediation, pre-effect information, rejection-stuttering, and state-consistency assumptions required for safety preservation.
4. It replaces a universal scalar reliability score with a support-aware assurance profile whose rates carry explicit numerators, denominators, confidence intervals where appropriate, and N/A semantics for unsupported quantities.
5. It defines AREF as a claim-dependent assurance evidence profile layered on telemetry rather than a competing transport, and the companion artifact includes a machine-readable draft schema plus conformance/negative cases.
6. It maps the framework to the pinned AEGISRUN v1.0.1 implementation, uses a separate Python implementation to reproduce its simple cryptographic golden vector, reruns the controlled research artifact, and reports implementation boundaries that prevent stronger signer-authentication, hard-resource-preservation, crash-atomic completeness, or live model-effectiveness claims.

2 Positioning and Related Work

2.1 Behavioral reliability, traces, and benchmark validity

Rabanser et al. ask how consistently, robustly, predictably, and safely agents behave over repeated trials [17]; their ICML 2026 framework is explicitly empirical. HAL provides standardized infrastructure and large-scale rollout analysis [8]. τ -bench evaluates tool-agent-user interactions and repeated-run consistency [28], while τ^2 -bench extends conversational interaction to a dual-control setting [1]. ToolEmu [19] and AgentDojo [5] expose risky tool use and prompt-injection failures. Log-analysis work shows that final outcomes can conceal scaffold failures, benchmark artifacts, and unsafe actions [9]. Trace Integrity argues that a correct final answer can be backed by an invalid structured computation and motivates execution contracts for auditable data-agent traces [6]. These works reinforce the need for process evidence. Our question is complementary: after choosing a property and an observation/control interface, which claims about an execution are actually decidable, enforceable, or auditable at that interface?

2.2 Closest theoretical antecedents

The observation-class construction used here has important predecessors and should not be read as originating equivalence-class uncertainty. Pawlak’s rough-set framework defines lower and upper approximations of a set relative to indiscernibility classes [29]; for a property P and observation relation, our certified region corresponds to the lower

Table 1: Closest-work boundary for the formal core. “Adds here” denotes the scoped integration in this paper, not a claim that the antecedent idea is absent from all prior work.

Work	Overlap adopted here	Adds here	Remaining boundary
Pawlak [29]	equivalence/indiscernibility classes; lower/upper approximations	execution-evidence interpretation and linkage to enforcement/evidence integrity	class-based certainty is established prior art
Bauer et al. [30]	conclusive vs. inconclusive monitoring verdicts	arbitrary evidence map over agent executions; claim-oriented audit semantics	finite-prefix temporal monitoring is a different setting
Cimatti et al. [31]	compatible runs under partial observation; out-of-model handling	evidence-profile integration and agent-gateway mapping	their framework supplies algorithms for a temporal-system setting
Schneider [21]	safety enforcement by mediated execution	typed Propose–Verify–Commit specialization with evidence obligations	safety gating itself is established
Aceto et al. [32]	sound approximation when exact monitoring is unavailable	explicit claim/evidence boundary for agent audit profiles	branching-time monitorability differs from this trace-evidence abstraction

approximation, the violated region to the complement of the upper approximation, and the unknown region to their boundary. Bauer, Leucker, and Schallhart formalize true/false/inconclusive runtime verdicts for finite observations of temporal properties [30]. Cimatti, Tian, and Tonetta monitor partially observable systems by reasoning over runs compatible with the observed execution and include an explicit assumption-violation/out-of-model outcome [31]. Aceto et al. study optimal sound monitorable approximations when exact monitoring is unavailable [32]. These results establish that three-valued uncertainty, compatible-run reasoning, and approximation under limited observation are prior foundations.

Our use of those foundations is therefore a specialization and integration, not a priority claim for the underlying mathematics. We model arbitrary evidence maps over complete agent executions, connect the resulting certification boundary to a pre-effect enforcement boundary and an integrity-preserving evidence contract, and test the mapping against a pinned tool gateway. Theorem 1 is included because it makes the paper’s information boundary explicit and reusable, not because its factorization idea is claimed as a new mathematical primitive.

2.3 Agent runtime enforcement and formal verification

Runtime verification treats traces as first-class objects and distinguishes offline verification, online monitoring, and enforceability [10]. Schneider’s security-automata result identifies safety properties as a central enforceable class under inline mediation [21]; Simplex illustrates runtime assurance around an uncertified high-performance controller [18]. Recent agent-specific systems occupy important parts of this design space. AgentSpec provides a DSL and runtime enforcement for agent constraints [25]; the version of Pro2Guard cited here is pinned to arXiv v1 and adds look-ahead risk estimation through probabilistic model checking [24]; Agent-C enforces temporal constraints on tool-call sequences [7]; Agent Behavioral Contracts formalize preconditions, invariants, governance, and recovery [2]; GuardAgent translates guard requests into executable checks [27]. VIGIL uses a verify-before-commit security architecture against tool-stream injection [11], and SafeMCP constrains capability acquisition in MCP ecosystems [26].

SABER is especially close at the systems boundary: it distinguishes mutating from non-mutating actions and introduces mutation-gated verification before consequential actions [4]. We therefore do not claim to originate mutation gating. Mercado and Lomuscio formalize stateful tool-enabled deployments over operational data and analyze FO-CTL verification, including undecidability and finite-domain conditions [12]. Their focus is system-level verification; ours is a claim-dependent evidence boundary coupled to a conditional effect gate and auditable record.

2.4 Concurrent agent-assurance and evidence systems

Several 2026 systems substantially overlap the integration and evidence layers of this paper and narrow the appropriate novelty claim. Nian et al.’s *Auditable Agents* distinguishes accountability, auditability, and auditing; organizes auditability around action recoverability, lifecycle coverage, policy checkability, responsibility attribution, and evidence

Table 2: Concurrent 2026 assurance/evidence systems and the remaining contribution boundary. “Partial” means the cited work addresses a neighboring mechanism but not the exact formal object used here.

Work	Formal execution claim	Pre-effect control	Evidence / trust semantics	Relationship to this paper
Auditable Agents [33]	auditability dimensions, not the observation-class iff criterion	enforce mechanisms include pre-execution mediation	lifecycle coverage, responsiveness, evidence integrity	broadest overlap with execution-assurance framing; this paper adds the explicit certifiability boundary and pinned source mapping
Paduraru et al. [34]	machine-checkable step/trace contracts	allow/rewrite/block runtime governance	deterministic replay, fault injection, trace metrics	strong overlap in trace assurance and mediation; this paper focuses on observation-induced decidability plus claim-dependent audit evidence
Brömme [35]	conceptual evidence-claim model	policy assessment rather than a typed effect theorem	explicit claims, assumptions, limitations; hash $\neq$ truth, signature $\neq$ authorization	closest conceptual neighbor to AREF; this paper binds evidence semantics to certifiability/enforcement and a gateway case study
NovaFabric [36]	no observation-class certification theorem	capture is primarily evidentiary rather than an agent policy gate	OTel-compatible capture, signatures, timestamps, Merkle log, replay, explicit TCB	strong systems overlap for portable audit-grade evidence; this paper adds property-relative certification/enforcement semantics
Agent Flight Recorder [37]	no comparable certifiability theorem	post-hoc audit focus	structured events, hash chains, Merkle proofs, external anchoring	close evidence-integrity neighbor; this paper additionally studies pre-effect control and observation-limited assurance

integrity; and analyzes detect/enforce/recover mechanisms, including pre-execution mediation and tamper-evident records [33]. Paduraru, Bouruc, and Stefanescu instrument multi-agent executions as message-action traces with machine-checkable step/trace contracts, deterministic replay, stress testing, fault injection, and runtime governance actions such as allow/rewrite/block [34].

Brömme’s evidence model is especially close to the semantics of AREF: it separates evidence claims from their trust assumptions and limitations, and explicitly warns that a hash does not establish semantic truth, a signature does not establish authorization, and an anchor does not establish capture completeness [35]. NovaFabric builds portable, replayable, tamper-evident execution evidence from OpenTelemetry-compatible traces, signed evidence bundles, Merkle structures, timestamps, and an explicit trusted-computing-base boundary; its stated contribution is integration rather than new cryptography [36]. Agent Flight Recorder similarly records structured agent actions and combines hash chaining, Merkle batching, and third-party-verifiable anchoring for long-horizon tool-using agents [37].

These works mean that neither “agent auditability,” trace contracts, pre-effect mediation, claim/evidence semantics, nor tamper-evident execution records are individually novel here. The scoped contribution is the particular coupling of an observation-bounded certifiability criterion, a typed committed-effect enforcement theorem, a claim-dependent evidence profile, and a source-audited gateway case study with explicit failure boundaries. Table 2 makes this boundary concrete.

2.5 Composition, governance, protocols, and telemetry

Agent Behavioral Contracts II develops finite-sample compositional certificates *without an independence assumption* and reports strong co-failure dependence in same-model handoffs [3]. We therefore keep multi-agent composition separate from the per-execution assurance profile and make no independence-based product claim.

NIST AI RMF and its GenAI profile provide risk-management vocabulary and lifecycle practices rather than a formal execution-certification calculus [23, 15]. MCP’s 2026-07-28 revision strengthens authorization and supplies a standardized tool interface, but protocol transport is not itself an assurance proof [13]. OpenTelemetry’s GenAI conventions moved in 2026 to a dedicated GenAI semantic-conventions repository; the relevant conventions remain a telemetry carrier rather than a cryptographic or semantic-completeness guarantee [16]. We consequently define AREF as a profile over assurance evidence semantics, not another telemetry transport.

Novelty boundary. The paper’s defensible contribution is combinational and case-study based. Established observation/monitoring principles are specialized to tool-using agent executions; concurrent 2026 work already covers broad agent auditability, trace contracts and governance, evidence-claim semantics, replayable signed execution evidence, and tamper-evident agent flight records [33, 34, 35, 36, 37]. What remains distinctive within the literature reviewed here is the particular coupling of (i) an observation-bounded Boolean-certifiability criterion and conservative audit semantics, (ii) a typed committed-effect enforcement theorem, (iii) a claim-dependent evidence profile with explicit trust boundaries, and (iv) a pinned source-level audit of a concrete gateway/evidence implementation. Stronger claims of a wholly new general certification theory, a new auditability paradigm, or novelty of tamper-evident agent logging are not made.

3 System and Threat Model

3.1 Execution traces

Let an agentic deployment consist of an untrusted proposer A , an environment E , tools U , a policy set K , an enforcement boundary G , and an evidence subsystem L . An execution is a finite or infinite trace

$$\tau = \langle e_0, e_1, \dots \rangle \in \mathcal{T},$$

where an event may contain state/input references, a proposed action $\hat{a}_t$, a prospective effect descriptor $\hat{u}_t$, a gate decision d_t , a committed action a_t , an observed tool result or effect receipt u_t , a policy reference, check outcomes, and evidence metadata. We do *not* assume access to hidden chain-of-thought. Natural-language rationales may be logged for debugging but are not privileged causal evidence.

A trace property is a set $P \subseteq \mathcal{T}$. Prefix-safety properties are those for which a violation has a finite bad prefix that cannot be repaired by extension. Examples include “never commit a tool call forbidden by the active policy” and “a rejected proposal causes no mediated external effect.” Bounded-liveness properties require progress within an explicit bound; unconstrained liveness generally cannot be guaranteed merely by blocking actions.

3.2 Observation boundary

Definition 1 (Observation map). *An observation map is a function $\text{Obs} : \mathcal{T} \rightarrow Y$ that maps a complete execution to the evidence available to an auditor.*

We use an illustrative hierarchy:

- Ω_0 : final outputs/outcomes only;
- Ω_1 : outputs plus tool/effect surface logs;
- Ω_2 : structured proposals, policy/check references, decisions, committed operations, observed results/effect receipts, and ordering;
- Ω_3 : Ω_2 plus integrity metadata sufficient to verify ordering/content binding and an authenticated evidence root.

The hierarchy is about information and integrity, not about a particular logging vendor.

3.3 Enforcement boundary and trusted computing base

Assumption 1 (Complete mediation). *Every externally consequential operation in scope passes through G before the operation is invoked. The agent cannot directly invoke the same effectful capability through an unmediated path.*

The trusted computing base (TCB) for an enforcement claim includes the policy evaluator, commit gate, effect executor interface, relevant recorder/integrity components, persistent store, signing-key handling, ordering/clock assumptions when used, and any external-tool behavior explicitly relied upon by the property. A gate cannot guarantee “no side effect” for an effect it cannot mediate.

3.4 Threats and non-goals

We consider stochastic failures, tool failures, prompt/tool-output injection, policy-bypass attempts, evidence deletion/modification/reordering, policy-snapshot substitution, and partial infrastructure failure. We do not claim

Table 3: Core execution-assurance properties. “Conditional” means the result requires the stated observation and TCB assumptions.

Property	Meaning	Min. obs.	Monitor	Enforce	Audit/certify	Principal assumptions
ARI-CP	Committed actions preserve declared policy constraints	Ω_2	yes	conditional	conditional	policy decidable at gate; complete mediation; modeled effects sufficient
ARI-NSE	Rejected mediated proposals cause no external effect	Ω_2	yes	conditional	conditional	executor invoked only after allow; no bypass
ARI-TC	Every in-scope proposal/decision/commit has required evidence	Ω_2	yes	partial	conditional	durable-record semantics; crash model explicit
ARI-RB	Resource budgets (calls/time/retries/egress) are not exceeded by committed work	Ω_2	yes	conditional	conditional	counters complete; resource boundary authoritative
ARI-CC	Execution is bound to the intended immutable policy version	Ω_2	yes	conditional	conditional	stable policy identifier/hash and binding
ARI-EI	Preserved evidence has not been altered after its authenticated root was produced	Ω_3	yes	n/a	conditional	hash/signature assumptions and key authenticity
ARI-RS	Elevated model/runtime risk is surfaced before commitment	Ω_2	if signal exists	conditional	conditional	a defined risk functional and trusted signal source

universal alignment, correctness of arbitrary natural-language goals, truthful hidden reasoning, immunity to a compromised TCB, or that a signed record proves the recorded event was truthful at ingestion.

4 Execution Assurance Properties

The original invariant catalog is retained only where the semantics are externally testable. Table 3 separates monitorability, enforceability, auditability, and certifiability rather than treating them as synonyms.

ARI-RS is deliberately optional in the reference implementation because AEGISRUN v1.0.1 has no first-class generic risk-signal field. Likewise, ARI-RB refers to concrete resource bounds; a tool-call budget is not a proof of structural recursion depth.

5 Observability and the Certifiability Frontier

5.1 Observation equivalence

Definition 2 (Observational equivalence). For $\tau_1, \tau_2 \in \mathcal{T}$,

$$\tau_1 \sim_{\text{Obs}} \tau_2 \iff \text{Obs}(\tau_1) = \text{Obs}(\tau_2).$$

The observation class of τ is $[\tau]_{\text{Obs}} = \{\tau' \in \mathcal{T} : \tau' \sim_{\text{Obs}} \tau\}$.

An observation class is *mixed* for property P if it contains both a trace in P and a trace outside P .

Let $Y_{\text{real}} = \text{Obs}(\mathcal{T})$ denote the realizable observation domain.

Definition 3 (Three-valued certification semantics). For $y \in Y_{\text{real}}$, let $C_y = \text{Obs}^{-1}(y)$, which is nonempty by construction. Define

$$V_P^{\text{Obs}}(y) = \begin{cases} \text{CERTIFIED} & C_y \subseteq P, \\ \text{VIOLATED} & C_y \cap P = \emptyset, \\ \text{UNKNOWN} & \text{otherwise.} \end{cases}$$

An operational verifier receiving $y \notin Y_{\text{real}}$ returns `OUTOFMODEL` (or another explicitly invalid-input outcome) rather than treating the empty compatible set as a positive certificate.

This distinguishes positive certification from violation proof, insufficient evidence, and model/input invalidity. A conservative implemented verifier may return `UNKNOWN` even on a pure class; it must not return `CERTIFIED` when a compatible violating trace exists, nor `VIOLATED` when a compatible satisfying trace exists.

For a Boolean decision function $B : Y_{\text{real}} \rightarrow \{\text{PASS}, \text{FAIL}\}$, B is *sound* for P if $B(\text{Obs}(\tau)) = \text{PASS}$ implies $\tau \in P$, and *complete* if $\tau \in P$ implies $B(\text{Obs}(\tau)) = \text{PASS}$.

Theorem 1 (Boolean certifiability under observation equivalence). *There exists an observation-only Boolean decision function $B : Y_{\text{real}} \rightarrow \{\text{PASS}, \text{FAIL}\}$ that is both sound and complete for P iff P is constant on every $\sim_{\text{Obs}}$ equivalence class; equivalently, P is a union of observation classes.*

Proof. If a mixed class contains $\tau_1 \in P$ and $\tau_2 \notin P$ with identical observation y , any Boolean $B(y)$ gives the same answer for both traces and is therefore either unsound or incomplete. Conversely, if each realizable observation class is pure, define $B(y) = \text{PASS}$ exactly when $C_y \subseteq P$. Because C_y is nonempty for $y \in Y_{\text{real}}$, this is well-defined, sound, and complete. $\square$

Information-theoretic scope. Theorem 1 is a set-theoretic existence characterization, not an algorithm-existence or complexity result. Even with injective observation, choosing P to encode an undecidable language yields pure classes without yielding a computable total verifier. Executable certification therefore additionally requires an effective representation of observations and a decidable property language, or a sound approximation that may return `UNKNOWN`.

Corollary 1 (Mixed classes require uncertainty). *Every three-valued verifier that is sound for both positive and negative certification must return `UNKNOWN` on each mixed realizable observation class.*

Definition 4 (Observation refinement). Obs_2 *refines* Obs_1 if there exists f with $\text{Obs}_1 = f \circ \text{Obs}_2$. Thus Obs_2 partitions trace space at least as finely as Obs_1 .

Corollary 2 (Monotonicity under refinement). *If P is Boolean-certifiable under Obs_1 and Obs_2 refines Obs_1 , then P is Boolean-certifiable under Obs_2 for the same trace universe, property, and trust assumptions.*

Proof. Every Obs_2 class is contained in an Obs_1 class. Purity of all coarse classes therefore implies purity of all refined classes. $\square$

The *certifiability frontier* is property- and universe-relative: more telemetry is useful only when it splits mixed classes relevant to the property. Integrity metadata has a distinct role: it can establish that the observed record is bound to a trusted or authenticated root under specified key assumptions, but it cannot reconstruct an unrecorded latent fact by itself.

6 Runtime Enforcement

6.1 Propose–Verify–Commit

A mediated effectful step follows

$$\text{PROPOSE} \rightarrow \text{VERIFY} \rightarrow \text{COMMIT?} \rightarrow \text{RECORD}.$$

The agent is an untrusted proposer. Let $\pi_{\text{commit}}(\tau)$ project a full execution trace onto the finite sequence of committed in-scope external states/actions. A rejected proposal is a stuttering transition on this committed projection, although it may still create audit events in the full trace.

Let S be a prefix-closed set of safe finite committed prefixes. This typing matters: a property such as “every rejected proposal is durably recorded” is a full-trace evidence property, not a property of π_{commit} alone.

Theorem 2 (Prefix-safety preservation under complete mediation). *Assume: (i) the initial committed prefix lies in S ; (ii) every in-scope effectful transition is completely mediated by G ; (iii) before effect, G evaluates the actual next committed transition from a state consistent with the state that will be committed and rejects every extension outside*

S ; (iv) rejection causes no in-scope external effect and therefore stutters on π_{commit} ; and (v) concurrent admission is serialized, reserved atomically, or otherwise shown not to invalidate condition (iii). Then every committed prefix produced through G lies in S .

Proof. By induction over committed steps. The base prefix lies in S by (i). A rejected proposal leaves the committed projection unchanged by (iv). For an allowed proposal, conditions (iii) and (v) ensure that the transition checked by the gate is the transition applied to the relevant committed state; the extension therefore remains in S . Complete mediation (ii) excludes an unexamined in-scope effectful transition. $\square$

The theorem is intentionally conditional and specializes established safety-enforcement reasoning to the paper’s committed-execution boundary. If a tool can produce an effect not represented in the prospective model, the agent has a bypass capability, concurrent state changes invalidate admission, or an external effect occurs before durable evidence and a later transaction rolls back, different assurance properties fail for different reasons. Monitoring, enforcing, auditing, and certifying therefore remain separate columns in Table 3.

6.2 Assurance profile instead of a universal score

A single weighted value can hide qualitatively different failure modes and can be gamed by empty denominators or over-blocking. We report a profile

$$\mathbf{R} = (C_{cp}, T, G, V, F, U, I, L, S),$$

where components may include constraint-preservation rate C_{cp} , trace completeness T , gate coverage G , violation-prevention rate V , false-block rate F , unknown/certification coverage U , integrity-attack detection I , latency L , and task success S . Each empirical rate is reported with (n, d) support; $d = 0$ is N/A, never automatically 1. A deployment may define a utility projection over the profile, but the weights are a policy choice rather than a universal reliability definition.

7 AREF: An Assurance Evidence Profile

AREF is not a new transport. It is a claim-dependent evidence contract: a verifier should be able to determine which evidence fields, ordering rules, policy bindings, effect receipts, and trust anchors are required for the specific assurance claim being made.

Draft conformance contract. Revision 2.1 retains three field classes: **required-for-core** fields are necessary for any AREF event stream that claims policy-mediated execution; **required-for-claim** fields are activated by claims such as ARI-RB, ARI-RS, or ARI-EI; and **optional** fields carry debugging or explanatory context. The companion artifact contains a machine-readable draft schema `research/aref/aref-0.1.schema.json`, a validator, one conforming example, and negative cases.

Table 4: AREF v0.1 draft evidence rules.

Class	Examples	Rule
required-for-core	run/event id, sequence, action/tool id, policy ref+digest, decision, lifecycle state	missing core field invalidates AREF conformance for mediated-policy claims
required-for-claim	resource counters, risk signal, observed effect/receipt, integrity binding, signer trust id	mandatory only when the corresponding assurance claim is asserted
optional	natural-language rationale, raw prompt/payload, descriptive tags	never used as sole evidence for the core formal claims

Ordering and event quality. Sequence identifiers must be unique within a run and monotone under the declared ordering rule. Missing or duplicate core events are evidence defects, not silently ignored observations. A transport may deliver duplicates or out of order; the assurance layer must normalize them deterministically or return UNKNOWN/OUTOFMODEL for claims whose prerequisites cannot be reconstructed. Retries must have stable attempt identity. Policy references bind both a version/reference and content digest so a later policy replacement does not retroactively change what was evaluated.

Commit and effect semantics. The profile distinguishes proposed effects from observed results or effect receipts. An `allow` decision is not itself proof that an effect occurred; conversely, an effect receipt is not proof that the decision was authorized unless it is linked to the corresponding proposal, policy digest, and commit lifecycle. Evidence completeness claims require a declared coverage basis outside the evidence stream itself; a self-consistent log cannot prove that the recorder omitted nothing.

Integrity and signer identity. At an integrity-bearing level, event content/order and relevant policy/run outcomes are bound into an authenticated root. Signature verification establishes integrity relative to the verification key used. Authentication of the *intended recorder* additionally requires an external trust binding (for example a pinned key/fingerprint, certificate chain, or allowlisted signer identity); a public key supplied only by the bundle is not by itself that trust anchor.

OpenTelemetry mapping. OpenTelemetry’s GenAI semantic conventions are now maintained in a dedicated GenAI conventions repository [16]. Table 5 shows the intended layering. AREF-specific attributes/events should be namespaced until standardized; cryptographic chain/signature semantics and claim-specific completeness rules remain outside generic telemetry.

Table 5: Illustrative AREF-to-OpenTelemetry mapping.

AREF item	OTel carrier	Added assurance semantics
run id	trace/agent-run linkage	stable assurance identity
step/event id	spans/events	unique declared ordering
tool name	GenAI/MCP tool attributes	governed tool identity
proposal	event/attributes	pre-commit action representation
policy ref/hash	custom assurance attrs	immutable policy binding
decision	event/status	allow/block/redact/degrade/approval
observed effect/result	tool result/event	post-commit observation/receipt
prev/event hash	outside generic OTel	tamper-evident chain
signature/root + trust id	outside generic OTel	evidence attestation plus signer binding

Proposition 1 (Integrity is not truth). *Assuming collision resistance of the event/root hash and unforgeability of the signature under an authentic verification key, modification of evidence covered by the authenticated root is detectable after signing. This does not imply that the event was truthful when first recorded, that omitted events never occurred, that the signer is the intended recorder absent an external trust binding, or that the evidence remains available.*

This distinction is essential: “tamper-evident” is not “tamper-proof,” cryptographic consistency is not semantic completeness, and an embedded verification key is not automatically an authenticated identity.

8 Reference Implementation: AegisRun v1.0.1

We audited the public AEGISRUN repository at commit `bd41a343ad3263fa36da6cb241f03733d11e50ea` [20]. The audit is source-level rather than inferred from marketing documentation.

8.1 Strong correspondences

The primary tool-call path in `api/internal/gateway/gateway.go` loads the run’s versioned policy, compiles it, checks runtime budget state, evaluates the proposal, persists the tool-call request and decision, and only then dispatches according to the policy action. The `block` and current `require_approval` branches do not invoke the tool executor. This is a concrete Propose–Verify–Commit boundary for calls routed through the gateway.

The evidence subsystem persists per-run events with sequence numbers, previous hashes, and event hashes. A completed evidence bundle format 1.0.0 contains `manifest.json`, `events.jsonl`, `policy.snapshot.json`, `run.json`, an optional `public_key.pem`, and explanatory metadata. The offline verifier checks chain linkage/content, policy-snapshot consistency, and Ed25519 signatures when signing material is present. A golden-bundle test exercises chain, policy, and signature verification. Versioned policy references and content hashes provide a concrete binding for ARI-CC.

AEGIS also implements tool/argument policy rules, egress restrictions, redaction, runtime budgets, PostgreSQL persistence, Prometheus instrumentation, OpenTelemetry tracing, Python/TypeScript SDKs, OIDC/RBAC/tenant controls, and operational deployment assets. These are useful engineering controls but are not all formal contributions of this paper.

8.2 Gaps that bound paper claims

Table 6 records the most important audit conclusions.

Table 6: Theory-to-AEGIS mapping at the audited v1.0.1 commit.

Mechanism	Status	Evidence in current implementation	Consequence for paper claim
Pre-effect policy decision	Implemented	gateway evaluates policy before action switch/executor	strong match for mediated tool calls
Complete mediation	Assumption	central gateway exists, but bypass freedom is deployment-dependent	enforcement theorem remains conditional
Policy block $\Rightarrow$ no gateway tool execution	Implemented	block branch returns denial without executor call	supports ARI-NSE within gateway boundary
Trace completeness	Partial	ordinary request, decision, and response events persisted	budget rejection and crash paths prevent a universal completeness claim
Bundle integrity	Implemented	hash chain, policy snapshot, signed root, offline signature check	detects covered modification relative to the supplied/selected key
Intended-signer authentication	Conditional/partial	inspected verifier loads the public key from the bundle; no external signer allowlist was found in the inspected CLI/config	requires a pinned/externally authenticated trust anchor for recorder-identity claims
Approval/resume	Not implemented	<code>require_approval</code> decision returned; pending call/resume absent	no full asynchronous HITL claim
Hard resource preservation	Partial	call/time/retry/egress counters exist, but admission precedes the per-run transaction lock and some accounting occurs post-execution	strict ARI-RB requires atomic reservation/recheck or bounded-overshoot assumptions
Risk signaling	Not implemented as generic field	no first-class ARI-RS contract found	theoretical/optional only
Degradation	Partial	synthetic degraded response path	not a general verified fallback controller
OTel GenAI profile	Partial	generic gateway span with run/step/tool attrs	interoperability base, not full AREF/GenAI conformance

Crash-consistency boundary. An allowed external tool is invoked while the surrounding database transaction is still open. An external effect can therefore succeed before a later persistence failure rolls back database evidence. The implementation consequently does not justify “every external side effect has durable evidence.” Stronger crash-atomic trace completeness requires an effect protocol such as an idempotent command/outbox scheme, a durable pre-commit record plus reconciled effect receipt, or application-specific transaction/compensation.

Signer-trust boundary. The inspected bundle loader accepts `public_key.pem` from the evidence bundle and the verifier checks the signature against that key. This proves cryptographic consistency relative to the key being used, not that the key belongs to the intended recorder. Proposition 1 therefore retains the authentic-key assumption. A deployment claiming authenticated signer identity must bind the verifier to an external trust source.

Resource-admission boundary. Runtime budget checks occur before the per-run transaction lock, while some quantities—including serialized result-size accounting—are observed after tool execution. Two concurrent requests can therefore require an atomic reservation/recheck argument before strict “never exceed” resource preservation follows. Wall-time admission also does not by itself interrupt an already-running tool at the limit. We treat AEGIS resource controls as useful admission/accounting mechanisms and ARI-RB as partial unless those stronger assumptions are established.

Budget rejection and telemetry boundaries. Budget rejection can occur before the ordinary tool-call evidence path, so budget enforcement is stronger than budget-rejection trace completeness. AEGIS uses OpenTelemetry, but the audited gateway emits a generic tool-call span rather than the complete AREF contract proposed here.

9 Evaluation

Our evaluation separates four evidence types: (i) a live source/CI audit of AEGIS; (ii) deterministic formal/evidence experiments; (iii) exact cross-language conformance against AEGIS’s own Go golden vectors; and (iv) a pinned external-workload preflight. We do not report controlled parameters as production AEGIS safety rates, and we do not fabricate a model-provider-backed live run when provider credentials are absent.

9.1 RQ1: What does observation refinement change?

We construct 2,254 matched traces. Groups share final outputs and tool/effect surfaces but differ in one of six properties: constraint preservation (CP), no-side-effect-on-reject (NSE), trace completeness (TC), resource-bound compliance (RB), policy consistency (CC), and evidence integrity (EI). We project the same corpus through Ω_0 – Ω_3 and apply the three-valued semantics from Section 5.

At Ω_0 and Ω_1 , only 6.8% of traces fall into conclusive classes in this adversarially paired corpus; 93.2% remain UNKNOWN. Structured Ω_2 evidence makes CP, NSE, TC, RB, and CC fully separable in the corpus, while EI remains mixed because integrity state is intentionally withheld. Mean conclusive coverage across the six properties is 95.6% at Ω_2 . Ω_3 separates EI as well, producing conclusive classes for all six modeled properties. Figure 1 reports mean conclusive coverage. This controlled corpus illustrates Theorem 1 and Corollary 2 on the disclosed finite universe $\mathcal{T}_{\text{exp}}$; purity inside $\mathcal{T}_{\text{exp}}$ is not a deployment certificate, and a larger trace universe can merge the observed support with previously absent counterexamples.

9.2 RQ2: What exactly does authenticated evidence detect?

The artifact builds a compact AEGIS-compatible integrity fixture: chained SHA-256 event hashes, a policy-spec hash, an evidence root bound to the final event/policy/outcome, and an Ed25519 signature. Nine post-root attacks are rejected: event deletion, event-payload mutation, event reordering, policy-snapshot mutation, run-outcome mutation, public-key replacement, trace truncation, signature replacement, and a stronger forgery that mutates an event and recomputes the downstream chain and root without the signing key. Thus the harness detects 9/9 injected post-root corruption/forgery classes. The public-key-replacement fixture changes the key without a matching valid signature; it therefore tests signature consistency, not intended-signer authentication. A bundle rebuilt and re-signed under a replacement key would require an external trust anchor to distinguish the substitute signer.

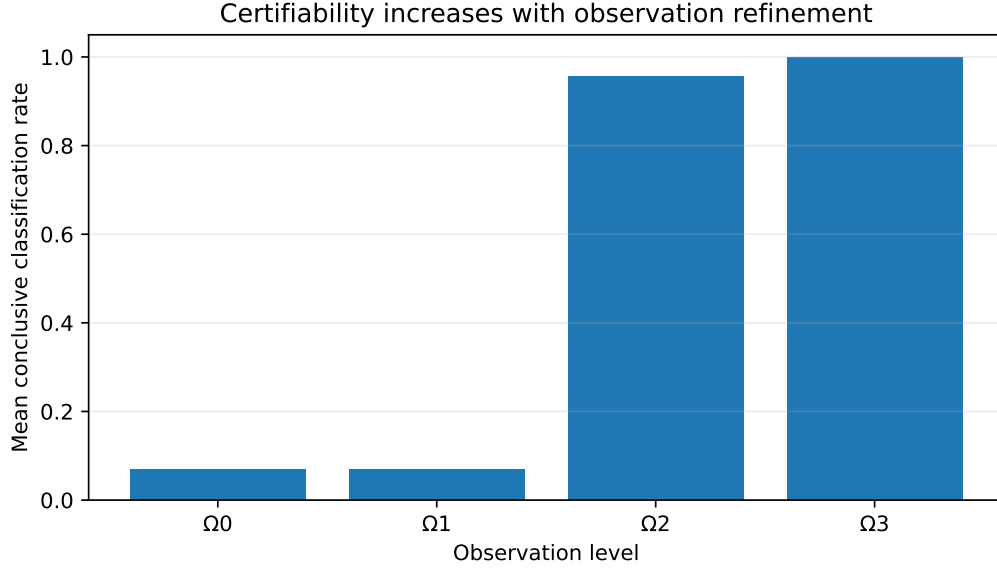


Figure 1: Observation ablation over the same 2,254 traces. Coverage rises only when added fields split property-relevant mixed classes.

A deliberately different negative control removes an event *before* root formation, recomputes the chain/root, and re-signs with the authorized key. The verifier accepts it. This is the expected result under a compromised recorder/signer and experimentally illustrates Proposition 1: authenticated integrity detects covered modification; it does not establish semantic completeness or recorder truthfulness.

To reduce the risk that the artifact merely approximates AEGIS’s cryptographic semantics, a separate Python conformance test reproduces the exact policy hash, two event hashes, evidence root, and Ed25519 verification constants from AEGIS’s Go `golden_bundle_test.go`; all five checks match at the pinned v1.0.1 commit.

9.3 RQ3: What does gating trade against?

We use a *paired* controlled proposal stream: the exact same 4,000 proposal truths are evaluated ungated and gated. The realized stream contains 1,176 violating and 2,824 permitted proposals. Ungated execution commits all 1,176 violations, an overall committed-violation rate of 29.4% (Wilson 95% CI: 28.0–30.8%). The illustrative gate is parameterized with a 98% violation-detection probability and 3% permitted-action false-positive probability. On the paired stream it blocks 1,161/1,176 violations and commits 15, for 98.72% prevention (95% CI: 97.91–99.23%); it falsely blocks 100/2,824 permitted proposals, a 3.54% false-block rate (95% CI: 2.92–4.29%). The remaining committed-violation rate is 0.375% of all proposals (95% CI: 0.227–0.618%). Figure 2 visualizes the trade-off.

The experiment validates metric definitions and support accounting; it does not establish a learned detector’s accuracy. SABER already provides model-facing evidence that mutating actions are disproportionately consequential and that mutation-gated safeguards can improve benchmark outcomes [4]. Our next systems experiment therefore targets the same consequential boundary while adding AEGIS policy/evidence measurements.

9.4 RQ4: What is the artifact-level verification/evidence cost?

The verifier timing protocol uses 1,000 warm-up verifications followed by 20 batches of 500 measurements (10,000 measured verifications) with Python garbage collection disabled during timed batches. Across three repetitions on the artifact-build environment (Python 3.13.5, cryptography 46.0.4, Linux x86-64, five logical CPUs reported to the container), mean verification time ranged from 0.211 to 0.218 ms, p95 from 0.299 to 0.327 ms, and p99 from 0.623 to 0.651 ms. These values are explicitly environment-specific and isolate the self-contained evidence verifier; they are *not* production AEGIS gateway latency.

For the same five-event fixture, serialized event JSONL occupies 1,511 bytes (302.2 bytes/event), and the constituent research-fixture fields total 2,032 raw bytes. Compressed archive size is not treated as a stable benchmark because

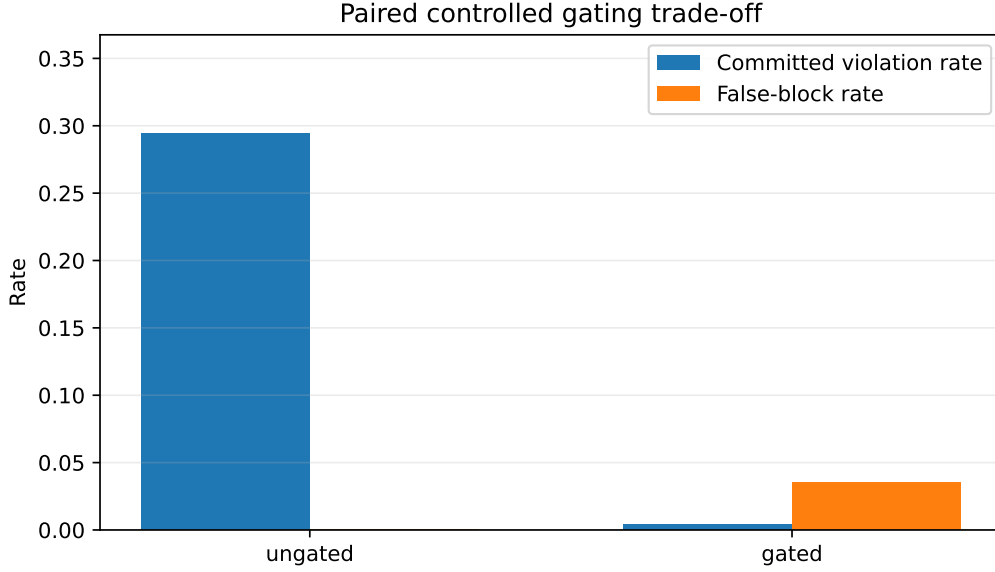


Figure 2: Paired controlled gating experiment. Detector probabilities are harness parameters, not measured production AEGIS safety rates.

fresh Ed25519 key and signature material changes DEFLATE compressibility slightly across reruns. These values are useful only as artifact-level accounting, not as a production storage forecast.

9.5 RQ5: Does a recognized workload actually exercise the effect boundary?

We preflight the current public τ airline workload rather than selecting it only by reputation. At the pinned 2026-09-17 repository commit, the 50-task airline file contains 142 expected actions. Forty-nine expected actions are on five environment-mutating tools (booking, cancellation, flight update, baggage update, passenger update), distributed across 26/50 tasks [22]. The workload therefore has substantial support for a future pre-side-effect gating study.

We also ingest an independently generated public five-task airline trajectory sample (Claude Sonnet 4.5 agent, GPT-4.1 user simulator) through the artifact adapter [14]. It contains 31 assistant tool calls and 4/5 successful tasks, but all 31 calls are reads/status checks or human transfer; there are zero mutating calls. We consequently reject this sample as evidence for gating effectiveness. This negative preflight result is scientifically useful: an agent benchmark name alone is insufficient; the realized trajectory corpus must contain the action class targeted by the assurance mechanism.

The artifact includes a pinned live protocol that routes each τ tool proposal through AEGIS and specifies paired seeds, complete-mediation checks, task success, prevention/false-block metrics, p50/p95/p99 latency, throughput, evidence bytes, failure injection, and raw-evidence publication. Executing that model-facing protocol requires provider/model credentials and is intentionally not represented as completed in this paper.

9.6 Reference-implementation validation

At the audited SHA, GitHub CI run 36104639864 and Security Scan run 36104639796 both succeeded on 25 September 2026. This is dated historical evidence, not a perpetual claim that later scheduled scans remain green. A scheduled security scan at the same SHA failed on 2 October 2026 and is not interpreted here without a causal investigation. CI status is not substituted for the paper’s formal assumptions.

10 Composition and Multi-Agent Systems

A universal weakest-link formula does not follow from an arbitrary weighted assurance score. We therefore avoid statements such as $R(A_{1:n}) \leq \min_i R(A_i)$ unless R is explicitly defined as a probability of a serial mission-success event under assumptions that make the inequality meaningful.

For multi-agent deployments, the execution-assurance model composes most cleanly at shared effect boundaries: each agent may propose actions, but an authoritative gate mediates externally consequential commits under a common policy/evidence contract. Local monitors can improve coverage but also introduce false blocks, latency, and failure modes. A watchdog therefore does not monotonically improve an arbitrary reliability metric without assumptions about detector quality, enforcement coverage, and its own availability.

Recent compositional evidence makes restraint especially important: Agent Behavioral Contracts II reports strong co-failure dependence in same-model handoffs and develops finite-sample certificates without an independence assumption [3]. We therefore do not multiply marginal component reliabilities. A useful future formalization for execution assurance is an assume-guarantee contract over agent boundaries: component i guarantees a trace property P_i provided its incoming evidence satisfies assumption Q_i ; system-level results then depend on whether upstream guarantees imply downstream assumptions and on explicitly measured dependence where probabilistic claims are made.

11 Limitations and Validity Threats

Observation models are abstractions. Real telemetry can be missing, delayed, duplicated, reordered, sampled, redacted, or semantically inconsistent. The formal results hold for the declared **Obs** and trace universe; deployment claims require showing that instrumentation implements that map and that the deployment universe is appropriately covered. A finite corpus can illustrate class purity without proving that no unseen counterexample exists.

Set-theoretic certifiability is not computability. Theorem 1 characterizes when a Boolean decision function exists on observation classes. Executable verification additionally depends on decidable property languages, effective state representations, algorithms, and resource bounds.

Complete mediation and state-consistent admission are deployment properties. A gateway’s existence does not prove that agents, plugins, credentials, shell access, or side channels cannot bypass it. Concurrent state changes can also invalidate a pre-effect decision unless admission is serialized or reserved atomically.

Prospective effects can be wrong. Pre-effect checks reason over what is known before execution. A nominally safe call may have unexpected effects due to a compromised tool or environment. Such guarantees require additional tool contracts or post-effect detection/compensation.

Evidence integrity is not semantic truth or signer identity. Hash chains and signatures detect covered post-hoc modification under cryptographic assumptions. They do not make a dishonest recorder truthful, recover omitted pre-signing events, or identify the intended signer when the verification key is accepted only from the same replaceable bundle. Authentication requires an external trust binding.

AEGIS implementation boundaries. The audited release lacks pending tool-call approval/resume; risk signaling is not generic; generic OTel tracing is not the full AREF mapping; budget rejection can precede normal decision-event persistence; external effects are not crash-atomically coupled to evidence persistence; strict hard resource preservation is not established under concurrent admission; egress accounting is post-effect; and wall-time admission does not prove in-flight interruption.

Evaluation scope. The 2,254-trace observation experiment is a constructed finite universe used to illustrate the observation-class characterization. The paired gating experiment samples a gate using chosen detection/false-positive probabilities and therefore validates metric/support accounting rather than detector effectiveness. The integrity harness is an enumerated fixture suite, not a universal cryptographic security proof. The evidence microbenchmark

is environment-specific and not production gateway latency. The pinned τ workload audit and external trajectory ingestion establish action support and parser behavior but are not a live AEGIS-gated model run.

Artifact reproduction and external archival location. The companion archive used for Revision 2.1 contains the experiment runner, raw results, figures, mapping audit, and reproduce script, and its invariant checks pass when rerun. The archival artifact is identified by DOI 10.5281/zenodo.23118665 (Zenodo). The DOI identifies the exact public archival release used by this manuscript. The Zenodo file record and release manifest provide the outer archive checksum; internal file hashes are recorded in SHA256SUMS.

Privacy and evidence minimization. More observation can increase certifiability but also increase privacy and security exposure. AREF should prefer stable references, digests, redacted fields, and opt-in content when full payloads are not required for the claimed property. The certifiability frontier therefore interacts with a data-minimization frontier.

12 Discussion

The formal separation among monitorable, enforceable, auditable, and certifiable properties changes how agent safeguards should be evaluated. A runtime guard may have excellent violation-detection recall yet be unable to prevent effects because it sits after the tool. A pre-effect gate may prevent a policy class but provide poor independent auditability if rejected proposals are not durably recorded. Rich telemetry may explain a failure while still being insufficient for positive certification because satisfying and violating traces collapse into the same observation class. Cryptographic integrity can protect the record without proving the record’s semantic completeness.

This perspective also clarifies the relationship to empirical reliability. Behavioral evaluation tells us how often an agent fails under repeated tasks, perturbations, or adversarial conditions. Execution assurance tells us which failures can be detected, prevented, or certified at a system boundary and under which assumptions. Neither substitutes for the other. In deployment, the strongest architecture combines both: benchmark and red-team the agent’s behavior, mediate dangerous capabilities, preserve evidence sufficient for the intended claims, and make unsupported conclusions explicitly UNKNOWN.

13 Conclusion

Agentic systems require assurance claims that are explicit about what was observed, what was mediated, and what trust assumptions bind the resulting evidence. This paper specializes established equivalence-class, partial-observability, runtime-monitoring, and safety-enforcement ideas to tool-using agent executions and connects them to a claim-dependent evidence profile and a pinned gateway implementation. On realizable observations, mixed observation classes cannot support sound-and-complete Boolean judgments, so uncertainty is a legitimate outcome rather than a verifier defect; executable certification additionally requires computable representations and decidable checks. On the enforcement side, complete mediation, pre-effect information, rejection stuttering, and state-consistent admission preserve committed prefix safety under the theorem’s explicit assumptions.

AREF turns those boundaries into an evidence contract rather than another telemetry transport. The AEGIS audit shows both the practical value and the limits of the integration: policy gating, version binding, hash-chained evidence, and signature checks are concrete, while signer authentication, strict concurrent resource preservation, crash-atomic completeness, asynchronous approval/resume, and live model-facing effectiveness require additional mechanisms or evidence. Controlled experiments and the rerun companion artifact support the paper’s scoped claims without converting simulation parameters or finite-corpus separability into production guarantees.

The central distinction remains: *behavioral reliability asks how agents perform and fail; execution assurance asks which claims about those executions are supportable or preventable under declared observation, control, and trust boundaries*. The value of the framework is therefore not a universal score or an unrestricted novelty claim, but a disciplined way to state where an assurance claim starts, what evidence makes it credible, and where it must stop.

A Reproducibility Checklist

The controlled-experiment source archive from which Revision 2.1 was hardened was `Agentic_Execution_Assurance_SOTA_2026_FINAL_artifact.zip`, with SHA-256

2bc2d86712785fd181a474c534c97b7ca27b66d19b720646dac87dae72c159b7.

The exact archival Revision 2.1 reproducibility package is identified by DOI 10.5281/zenodo.23118665. It contains the deterministic experiment runner, raw CSV/JSON results, generated figures, live AEGIS mapping audit, Go-to-Python golden-vector conformance, τ trajectory adapter, pinned airline-task structural audit, external real-model trajectory preflight, model-facing live-benchmark protocol, AREF v0.1 conformance material, and release metadata. The principal controlled-experiment seed is 20260925. The source audit pins AEGIS to commit `bd41a343ad3263fa36da6cb241f03733d11e50ea` and evidence format 1.0.0; the τ workload audit pins commit `b7ea9074c1cba482b30687fecdb5c8425fd6f619`.

For Revision 2.1, the archived `research/reproduce.sh` was executed again and its invariant checks passed: 2,254 traces, 9/9 post-root integrity fixtures detected, the authorized-recorder omission/re-sign control accepted as expected, AEGIS golden-vector conformance passed, and the public trajectory adapter reproduced 31 tool calls with zero mutating calls and four successful tasks. Timing measurements vary with the execution environment and are therefore not required to numerically match the original microbenchmark range. This is a clean rerun of the supplied artifact, not an independent external replication.

The Revision 2.1 package additionally contains `research/aref/aref-0.1.schema.json`, `research/aref/validate_aref.py`, and positive/negative conformance examples. The archival identifier for the exact Revision 2.1 package is DOI 10.5281/zenodo.23118665; the DOI identifies the public Zenodo archival record.

B AREF v0.1 Draft Field Classes

Core fields are stable run/event identity, per-run sequence, action/tool identity, policy reference and digest, decision, lifecycle state, and claim declarations. Effect claims additionally require observed effect/result evidence where applicable. Resource-bound claims require relevant resource evidence. Risk-signal claims require the risk field. Integrity claims require content/order bindings plus signer/trust metadata sufficient for the stated identity claim. Natural-language rationales and raw prompts are optional and are not sufficient evidence for the core formal claims.

C Artifact Result Summary

Quantity	Result
Matched trace corpus	2,254 traces
Ω_0 mean conclusive coverage	6.8%
Ω_1 mean conclusive coverage	6.8%
Ω_2 mean conclusive coverage	95.6%
Ω_3 mean conclusive coverage	100%
Post-root corruption/forgery classes detected	9/9
Compromised-signer omission negative control	verifier passes (expected)
AEGIS Go golden-vector conformance	5/5 checks
Paired gate prevention (95% CI)	98.72% (97.91–99.23%)
Paired gate false-block rate (95% CI)	3.54% (2.92–4.29%)
Verifier fixture mean range / p95 range	0.211–0.218 / 0.299–0.327 ms

References

- [1] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2 -bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.
- [2] Varun Pratap Bhardwaj. Agent behavioral contracts: Formal specification and runtime enforcement for reliable autonomous ai agents. *arXiv preprint arXiv:2602.22302*, 2026.

- [3] Varun Pratap Bhardwaj, Garima Singh, and Arun Pratap Bhardwaj. Agent behavioral contracts ii: Certifying compositional reliability without assuming independence. *arXiv preprint arXiv:2608.12895*, 2026.
- [4] Alejandro Cuadron, Pengfei Yu, Yang Liu, and Arpit Gupta. Saber: Small actions, big errors – safeguarding mutating steps in llm agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026.
- [5] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems*, 2024.
- [6] Srimonti Dutta and Akshata Kishore Moharir. Trace integrity for llm data agents: A vision for auditable structured reasoning in real-world systems. *arXiv preprint arXiv:2608.26036*, 2026.
- [7] Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. Enforcing temporal constraints for llm agents. *arXiv preprint arXiv:2512.23738*, 2025.
- [8] Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S. Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, et al. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.
- [9] Peter Kirgis, Sayash Kapoor, Stephan Rabanser, Nitya Nadgir, Cozmin Ududec, Magda Dubois, JJ Allaire, Conrad Stosz, Marius Hobbhahn, Jacob Steinhardt, and Arvind Narayanan. Log analysis is necessary for credible evaluation of ai agents, 2026. Accepted at the ICML 2026 FAGEN Workshop.
- [10] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.
- [11] Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. Vigil: Defending llm agents against tool-stream injection via verify-before-commit. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pages 9764–9785, 2026.
- [12] Alejandro J. Mercado and Alessio Lomuscio. Formal verification of agentic systems over operational data. *arXiv preprint arXiv:2608.03609*, 2026.
- [13] Model Context Protocol Project. Model context protocol specification, revision 2026-07-28, 2026. Released July 28, 2026.
- [14] Moyai AI. Public tau airline trajectory sample for agent-system analysis. <https://github.com/moyai-ai/tau2xagents-bench>, commit ebc4f9a44b7ff8427ea6b0a20d89a3158a32bdea, 2026. Trajectory file blob f0b1080b5fc81e2a6e0c7f8d01d999be0f559491.
- [15] National Institute of Standards and Technology. Artificial intelligence risk management framework: Generative artificial intelligence profile. Technical Report NIST AI 600-1, National Institute of Standards and Technology, 2024.
- [16] OpenTelemetry. OpenTelemetry semantic conventions for generative AI. Dedicated GenAI semantic-conventions repository, 2026. Accessed 2 October 2026; the GenAI conventions had moved from the core semantic-conventions repository.
- [17] Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of ai agent reliability. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.
- [18] Jose German Rivera, Alejandro Andres Danylyszyn, Charles Weinstock, Lui R. Sha, and Michael J. Gagliardi. An architectural description of the simplex architecture. Technical Report CMU/SEI-96-TR-006, Carnegie Mellon Software Engineering Institute, 1996.
- [19] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, 2024.
- [20] Stamatis-Christos Saridakis. Aegisrun v1.0.1: Policy-enforced tool gateway and tamper-evident evidence. <https://github.com/SaridakisStamatisChristos/AEGIS>, commit bd41a343ad3263fa36da6cb241f03733d11e50ea, 2026.

- [21] Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.
- [22] Sierra Research. tau benchmark repository: 2026-09-17 snapshot. <https://github.com/sierra-research/tau2-bench>, commit b7ea9074c1cba482b30687fecdb5c8425fd6f619, 2026. Airline task file blob ea4ff5e3f5e3d97ca391846a6e8dd9eb3437dc80.
- [23] Elham Tabassi. Artificial intelligence risk management framework (ai rmf 1.0). Technical Report NIST AI 100-1, National Institute of Standards and Technology, 2023.
- [24] Haoyu Wang, Chris M. Poskitt, Jun Sun, and Jiali Wei. Pro2guard: Proactive runtime enforcement of llm agent safety via probabilistic model checking. *arXiv preprint arXiv:2508.00500v1*, 2025. Version 1 pinned because later versions use the title ProbGuard.
- [25] Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable llm agents. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering*, pages 2938–2950, 2026.
- [26] Lichao Wang, ZhaoXing Ren, Tianzhuo Yang, Jiaming Ji, Chi Harold Liu, Yaodong Yang, and Juntao Dai. Safemcp: Proactive power regulation for llm agent defense via environment-grounded look-ahead reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pages 11374–11396, 2026.
- [27] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.
- [28] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025.
- [29] Zdzisław Pawlak. Rough sets. *International Journal of Computer & Information Sciences*, 11(5):341–356, 1982. doi:10.1007/BF01001956.
- [30] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14:1–14:64, 2011. doi:10.1145/2000799.2000800.
- [31] Alessandro Cimatti, Chun Tian, and Stefano Tonetta. Assumption-based runtime verification with partial observability and resets. In *Runtime Verification (RV 2019)*, LNCS 11757, pages 165–184. Springer, 2019. doi:10.1007/978-3-030-32079-9_10.
- [32] Luca Aceto, Antonis Achilleos, Adrian Francalanza, Anna Ingólfssdóttir, and Karoliina Lehtinen. The best a monitor can do. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, LIPIcs 183, pages 7:1–7:23, 2021. doi:10.4230/LIPIcs.CSL.2021.7.
- [33] Yi Nian, Aojie Yuan, Haiyue Zhang, Jiate Li, and Yue Zhao. Auditable agents. *arXiv preprint arXiv:2604.05485*, 2026.
- [34] Ciprian Paduraru, Petru-Liviu Bouruc, and Alin Stefanescu. A trace-based assurance framework for agentic ai orchestration: Contracts, testing, and governance. In *Proceedings of the 21st International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pages 420–427. SciTePress, 2026. doi:10.5220/0014840300004015.
- [35] Arslan Brömme. An evidence model for agentic processes: Evidence claims, trust assumptions, and policy assessment. *arXiv preprint arXiv:2609.08481*, 2026.
- [36] Mohsen Seyedkazemi Ardebili. Novafabric: Tamper-evident, replayable evidence for autonomous ai agent runs. *arXiv preprint arXiv:2609.12582*, 2026.
- [37] Laurent Bindschäedler, Quentin Botha, and Christoph Siebenbrunner. Agent flight recorder: Tamper-evident audit trails with on-chain anchoring for long-horizon tool-using agents. *arXiv preprint arXiv:2609.01931*, 2026.