

Reconstructive Authority Model: Runtime Execution Validity Under Partial Observability

Agent Governance Series — Paper 5

Marcelo Fernandez

Independent Research

DOI: 10.5281/zenodo.[TBD]

April 2026

Paper	Short title	Zenodo DOI	arXiv
P0	Atomic Decision Boundaries [1]	10.5281/zenodo.19642166	TBD
P1	Agent Control Protocol (ACP) [2]	10.5281/zenodo.19642405	arXiv:2603.18829
P2	From Admission to Invariants (IML) [3]	10.5281/zenodo.19643761	TBD
P3	Fair Atomic Governance [4]	10.5281/zenodo.19643928	TBD
P4	Irreducible Multi-Scale Governance [5]	10.5281/zenodo.19643950	TBD
P5	Reconstructive Authority Model (RAM)	<i>this paper</i>	TBD

Abstract

Autonomous systems increasingly operate under partial observability where the true execution-relevant state $S_r(t)$ is never fully accessible. Existing governance mechanisms—trusted execution environments, oracle-signed state proofs, cryptographic attestation—enforce the integrity of computation and state projections. We show this is structurally insufficient: *an authenticated projection of state is necessary, never sufficient*.

We separate integrity from coverage: authenticated projection is necessary, never sufficient. The *Reconstructive Authority Model* (RAM) introduces a reconstruction gate that reasons over an explicit *coverage envelope*—comprising the proven state, declared assumptions, and the unobservable residual whose existence is established by Paper 2 (IML, [3])—and permits execution only when coverage is adequate for that action class; otherwise it narrows privileges dynamically or fails closed. Attestation proves trust in measurement, not completeness of execution-relevant reality.

We formalize RAM, prove its necessity via two theorems and three corollaries, present a hybrid RAM + Attestation architecture with privilege-narrowing, and demonstrate through case study and synthetic experiments that RAM eliminates invalid execution under all coverage levels while attestation-based systems exhibit invalid execution rates proportional to $(1 - |S_p|/|S_r|)$.

This paper is Paper 5 of the Agent Governance Series. It operationalizes the observability impossibility of Paper 2 (IML, [3]) and completes the execution semantics of the four-layer architecture of Paper 4 ([5]).

Contents

1	Introduction	4
2	Background	5
2.1	Attestation-Based Governance	5
2.2	Partial Observability	5
2.3	Agent Governance Series	5
3	Limitations of Attestation-Based Governance	6
3.1	Integrity Does Not Imply Semantic Correctness	6
3.2	The Self-Justification Boundary	6
3.3	External State Proofs and the Coverage Problem	6
3.4	Drift Outside the Proof Boundary	7
3.5	Enforcement Without Reconstruction	7
3.6	Formal Limitation	7
4	Reconstructive Authority Model	7
4.1	Core Principle	7
4.2	Positioning Against Related Approaches	8
4.3	Coverage Envelope	8
4.4	Formal Definition	8
4.5	Non-Persistence of Authority	9
4.6	Invariants as Constructive Conditions	9
4.7	Partial Observability Handling	9
4.8	Drift Interpretation	10
4.9	Execution Model	10
5	Theoretical Results	10
5.1	Theorem 1: Limits of Attestation Without Reconstruction	10
5.2	Theorem 2: Necessity of Reconstruction	11
6	Contrast: RAM vs Attestation-Based Models	12
7	System Architecture	13
7.1	Component Overview	13

7.2	Execution Flow	13
7.3	Failure Handling Model	14
7.4	Where Attestation Still Matters	14
8	Case Study: Autonomous Financial Transfer	14
8.1	Scenario	14
8.2	Attestation-Based System at t_1	15
8.3	Oracle-Extended System at t_1	15
8.4	RAM-Based System at t_1	15
8.5	Comparative Outcome and Edge Case	15
9	Experimental Evaluation	16
9.1	Environment Model	16
9.2	Reproducibility	16
9.3	Drift Injection Model	16
9.4	Metrics	17
9.5	Authority Reconstruction Function (Pseudocode)	17
9.6	Results	18
9.7	Key Findings	19
9.8	The Security–Execution Trade-off (OCR)	20
10	Implications for System Design	20
11	Discussion	21
11.1	Convergence with Attestation Under High Coverage	21
11.2	Limitations	21
11.3	Future Work	22
12	Relation to Governance Series	22
13	Conclusion	23
A	Formal Proof of Theorem 5.1	24

1 Introduction

Autonomous agent systems face a fundamental tension: decisions must be made at admission time based on available state, but execution happens later, when conditions may have changed. Existing governance frameworks address this through *attestation*—cryptographic mechanisms that verify consistency between the current execution environment and a previously approved state. Trusted Execution Environments (TEEs), oracle-signed state proofs, and justification-based control systems all operate on this principle.

The limitation of attestation-based governance is structural, not implementational. We **separate integrity from coverage**: attestation resolves the former but is silent on the latter. Integrity—that a computation was performed correctly on the observed state—can be cryptographically guaranteed. Coverage—that the observed state was *sufficient* to justify the action in the first place—cannot.

An authenticated projection of state is necessary, never sufficient. Since the provable state $S_p \subsetneq S_r$ strictly in any real system, there always exists a component $\delta = S_r \setminus S_p$ that influences execution validity but lies outside the attested model. When δ becomes execution-critical—due to runtime drift, emergent conditions, or delayed observability—attestation produces a false positive: a system that *appears* consistent but *is not valid* to execute.

We formalize this limitation and introduce the **Reconstructive Authority Model (RAM)**: a governance framework that shifts the fundamental question from “*Is execution still consistent with what was approved?*” to “*Can authority still be constructed from what is true now?*” RAM reasons over a *coverage envelope* and permits action only when coverage is adequate for the specific action class—narrowing privileges when coverage is partial, failing closed when it is insufficient.

Contributions.

- (i) A formal characterization of the structural limitation of attestation-based governance systems, with proof that attestation correctness does not imply execution validity (Section 3).
- (ii) The RAM formal model: definitions, non-persistence theorem, invariant treatment, and partial observability handling (Section 4).
- (iii) Two theorems establishing (a) the limits of attestation without reconstruction and (b) the necessity of reconstruction for execution validity (Section 5).
- (iv) A RAM+Attestation hybrid architecture with formal failure handling (Section 7).
- (v) A case study and synthetic experimental evaluation comparing Attestation, Oracle, and RAM models under drift injection (Sections 8–9).
- (vi) An account of RAM’s position within the Agent Governance Series (Section 12).

Series context. The Agent Governance Series develops a complete formal theory of autonomous agent governance. Paper 0 [1] establishes atomic decision boundaries—the structural requirement for guaranteeing execution-time admissibility. Paper 1 (ACP) [2] implements enforcement via admission control at the policy boundary. Paper 2 (IML) [3] proves the epistemological limits of

local observability: an agent can never fully observe the state space relevant to its own invariants. Paper 3 [4] addresses distributive limits under strategy-proof mechanisms. Paper 4 [5] proves the irreducibility of the four-layer governance architecture. This paper (P5) closes the series by addressing the runtime question: given that observability is incomplete (P2) and the architecture is irreducible (P4), how should a system decide whether to execute at all?

2 Background

2.1 Attestation-Based Governance

Trusted Execution Environments (TEEs) such as Intel SGX [6] provide hardware-backed attestation: a guarantee that a computation runs on unmodified code within a tamper-resistant enclave. At the governance level, TEE-based systems combine attestation with justification-based control [7], where an agent generates a justification block J prior to execution, and a TEE verifies that execution proceeds only if J is consistent with attested state.

Oracle-extended systems improve coverage by augmenting the attested state S_p with externally-signed state proofs S_p^{oracle} , reducing the gap between S_p and S_r [8]. Architectures such as ATLAS [9] combine TEE attestation with oracle-anchored state proofs to increase the trustworthy observable surface. However, as we prove formally (Corollary 5.3), no finite proof system can achieve $S_p = S_r$ in a general system with partial observability and dynamic state: oracle extensions reduce but cannot eliminate the state gap.

Runtime verification (RV) approaches [10–12] monitor execution against formal specifications, typically through monitors that observe traces and raise alarms on specification violations. A critical distinction: RV monitors assume the monitored state is complete—that what they observe is sufficient to evaluate the specification. RAM relaxes this assumption explicitly. Where an RV monitor asks “*does the observed trace satisfy the property?*”, RAM asks “*is what we observe sufficient to determine whether execution should proceed at all?*” This makes RAM complementary to RV: RV governs *how* execution proceeds; RAM governs *whether* it may.

2.2 Partial Observability

Partial observability is a fundamental feature of real-world agent systems, studied extensively in the context of Partially Observable Markov Decision Processes (POMDPs) [13]. The POMDP framework models agents that receive observations rather than direct state access, maintaining belief distributions over the real state. RAM shares this epistemic humility but addresses a different problem: not optimal policy under uncertainty, but binary authority—whether the system is *permitted to act at all*.

2.3 Agent Governance Series

The governance series establishes a formal framework for agent systems under finite observability. Paper 2 (IML) [3] is most directly relevant here: it proves formally that for any governance architecture with finite memory and local observation, there exists a violation path that is undetectable until it has already caused a constraint breach. RAM operationalizes the response to this result:

since full observability is unachievable (IML), authority must be constructed conservatively from whatever is observable, with explicit failure when observation is insufficient.

3 Limitations of Attestation-Based Governance

We identify six structural limitations of attestation-based governance. These are not implementation failures; they are consequences of the gap between S_p and S_r .

3.1 Integrity Does Not Imply Semantic Correctness

Attestation mechanisms guarantee that a computation has not been tampered with and executes within a trusted boundary. They do not guarantee that the *content* of that computation is semantically correct or complete relative to the execution context.

When execution authority is derived from internally generated artifacts (justification blocks), the system is vulnerable to:

- Incomplete representations of state;
- Omitted constraints;
- Structurally valid but semantically insufficient reconstructions.

This leads to the first failure class: a justification is valid, attestable, and internally consistent, yet insufficient to support safe execution.

3.2 The Self-Justification Boundary

Even when attestation enforces integrity, if the system under control is responsible for generating its own justification artifacts, a structural dependency emerges:

Execution authority depends on the system's ability to correctly describe the conditions under which it is allowed to act.

This creates a failure mode where: (1) the system produces a formally valid justification; (2) the attestation verifies its integrity; (3) execution proceeds; yet the justification does not fully capture the relevant execution constraints. This is not a failure of *enforcement*, but of *epistemic completeness*.

3.3 External State Proofs and the Coverage Problem

To address self-justification, some architectures introduce externally signed state proofs (oracles, ledgers). While this improves trust, it introduces a new limitation: the system operates against a *provable projection* of reality, not reality itself.

No external proof system can guarantee complete coverage of all execution-relevant state, due to:

- **Observability limits:** not all variables are measurable;
- **Latency:** state may change between proof generation and execution;
- **Model constraints:** only predefined dimensions are captured.

Thus, even perfect comparison between internal intent and external proof can result in correct validation against an incomplete state model.

3.4 Drift Outside the Proof Boundary

Attestation-based systems typically detect divergence as mismatches between: (i) attested state at admission and (ii) current state at execution. However, divergence can also arise from state dimensions that were *never included* in the attested or provable model.

This creates a blind spot: no mismatch is detected, the system remains “within envelope”, execution continues—despite the fact that the true execution basis has shifted.

3.5 Enforcement Without Reconstruction

Attestation-based governance models are fundamentally *enforcement-oriented*: they verify consistency, enforce constraints, and terminate on mismatch. However, they do not inherently address whether execution authority can still be *constructed* from the current state.

A system may pass all attestation checks, remain within defined envelopes, and maintain cryptographic validity—and still lack a valid basis for execution under actual conditions.

3.6 Formal Limitation

Definition 3.1 (State gap). *Let $S_r(t)$ be the real execution-relevant state at time t , $S_p(t)$ the provable state accessible to attestation, and $J(t)$ the justification derived from $S_p(t)$. The state gap at time t is $\delta(t) = S_r(t) \setminus S_p(t)$.*

Under any attestation model, even when $J(t)$ is valid relative to $S_p(t)$, $S_p(t) \subseteq S_r(t)$, and all attestation checks pass, it does not follow that $J(t)$ is valid relative to $S_r(t)$.

Therefore: *attestation correctness does not imply execution correctness.*

4 Reconstructive Authority Model

4.1 Core Principle

Execution authority is not a persistent property granted at admission, but a derivable property that must be continuously reconstructible from the current state.

Execution is valid at time t if and only if authority can be constructed from $S_r(t)$. This removes dependency on historical decisions, stored justifications, and previously attested envelopes.

4.2 Positioning Against Related Approaches

RAM must be distinguished from two related but distinct concepts.

Conservative halt policies define a rule: “halt if uncertainty exceeds threshold θ .” This requires a probability measure over states and a tunable threshold. RAM does not require either: F is not a probabilistic function but a deterministic constructor that returns \perp when required state components are missing—not when uncertainty is high. RAM can halt with certainty (when F is definitively false) and can execute under uncertainty (when available components suffice to determine $F = \text{true}$).

POMDP-based abstention requires a belief state $b(s)$ over S_r and a value function $V(b)$ over beliefs. RAM requires neither: it operates on the observable state $\hat{S}_r(t)$ directly, returning \perp when authority is not constructible from what is available—a structural condition, not a probabilistic one.

The contribution of RAM is not “halt when uncertain” but: *define execution authority as a constructible property of the current state, such that execution is permitted only when authority can be derived.* This is a different problem from uncertainty quantification or belief-based decision-making.

4.3 Coverage Envelope

Definition 4.1 (Coverage Envelope). *The coverage envelope $\mathcal{E}(t)$ at time t is the triple:*

$$\mathcal{E}(t) = (S_p(t), \mathcal{H}(t), \delta(t))$$

where:

- $S_p(t) \subseteq S_r(t)$: the proven state—components that are observable and cryptographically attestable at time t ;
- $\mathcal{H}(t)$: declared assumptions—explicit propositions about unobserved state that the system accepts as operationally necessary for authority construction;
- $\delta(t) = S_r(t) \setminus S_p(t)$: the acknowledged unobservable residual—the gap whose permanent existence is established by Paper 2 (IML, [3]).

Coverage is adequate for action class α if the authority constructor can derive $A(t) = \text{true}$ from $\mathcal{E}(t)$ under the requirements of α . Coverage is partial for α if it suffices for a strict subset $\alpha' \subsetneq \alpha$. Coverage is insufficient if no $\alpha' \subseteq \alpha$ can be authorized.

Remark 4.1. Making $\delta(t)$ explicit in the envelope is not merely bookkeeping: it forces system designers to acknowledge what they do not know at each execution step, rather than silently assuming the provable state is complete. This is the operational consequence of IML’s impossibility result.

4.4 Formal Definition

Definition 4.2 (Reconstructive Authority). *Let $S_r(t)$ be the real execution-relevant state at time t , \mathcal{A} the space of action classes, $A(t)$ the execution authority at time t , and $F : \mathcal{E} \times \mathcal{A} \rightarrow \mathcal{A} \cup \{\text{false}, \perp\}$*

the authority construction function over coverage envelopes. Then:

$$A(t) = F(\mathcal{E}(t), \alpha)$$

where α is the requested action class.

- If $F(\mathcal{E}(t), \alpha) = \alpha$: execute at full scope;
- If $F(\mathcal{E}(t), \alpha) = \alpha' \subsetneq \alpha$: execute with narrowed privileges (α' only);
- If $F(\mathcal{E}(t), \alpha) \in \{\text{false}, \perp\}$: halt or defer.

The value \perp represents insufficient information to determine authority for any sub-class; false represents a definitive refusal.

The inclusion of \perp is critical: when the observable state is insufficient to establish authority, RAM mandates halting rather than defaulting to execution.

4.5 Non-Persistence of Authority

Unlike admission-based models, authority in RAM is non-persistent:

Proposition 4.1 (Non-persistence).

$$A(t_0) = \text{true} \not\Rightarrow A(t_1) = \text{true} \quad (t_1 > t_0)$$

even when no explicit policy violation is detected between t_0 and t_1 .

Authority must be recomputed at each execution step, not assumed from prior successful computations. This is the key operational difference from attestation-based models.

4.6 Invariants as Constructive Conditions

In RAM, invariants are not validation checks but *constructive requirements*.

Instead of verifying “invariant holds,” we define: $F(S_r(t))$ is defined only if invariants hold over $S_r(t)$. If invariants fail, F becomes undefined (\perp), authority cannot be constructed, and execution must halt.

This shifts the semantics from *reactive* (detect violation, then stop) to *constructive* (establish validity, then proceed).

4.7 Partial Observability Handling

RAM explicitly acknowledges that $S_r(t)$ is never fully observable. Authority construction therefore operates under bounded uncertainty with explicit incompleteness:

$$F(S_r(t)) = \perp \text{ if required components of } S_r(t) \text{ are unknown.}$$

The default behavior is **halt or defer**, not continue under partial justification. This is the conservative assumption established in Paper 2 (IML) [3] as the only safe default under finite observability.

4.8 Drift Interpretation

In attestation-based models, drift is defined as a detectable mismatch between states at different points in time. Under RAM, drift is not defined as a mismatch but as *inability to reconstruct authority from current conditions*:

$$\text{Drift at } t \equiv F(S_r(t)) \in \{\text{false}, \perp\}$$

This eliminates the need to define, detect, or classify drift explicitly. Instead, drift manifests automatically as an authority failure.

4.9 Execution Model

At each execution step, the RAM execution loop is:

- (1) Observe current state $\hat{S}_r(t) \subseteq S_r(t)$ (partial, bounded);
- (2) Attempt to compute $A(t) = F(\hat{S}_r(t))$;
- (3) If $A(t) = \text{true}$: proceed with execution;
- (4) If $A(t) \in \{\text{false}, \perp\}$: halt or re-resolve.

No execution step occurs without a fresh authority computation. Historical authority grants are not consulted.

5 Theoretical Results

5.1 Theorem 1: Limits of Attestation Without Reconstruction

Theorem 5.1 (Attestation insufficiency). *No attestation-based governance system can guarantee execution validity with respect to the real execution state S_r unless execution authority is reconstructible from $S_r(t)$ at runtime.*

Formal setup. Let $S_p(t) \subseteq S_r(t)$ be the provable state, $J(t)$ the justification derived from $S_p(t)$, $A_p(t) = G(S_p(t))$ the attestation-based authority decision for some deterministic function G , and $A_r(t) = F(S_r(t))$ the real authority decision. Execution validity requires $A_r(t) = \text{true}$.

Assumptions.

Assumption 5.1 (Partial observability). $S_p(t) \subsetneq S_r(t)$ in general (strict inclusion).

Assumption 5.2 (Attestation correctness). *All attestation mechanisms correctly verify the integrity of $J(t)$ and the authenticity of $S_p(t)$.*

Assumption 5.3 (Deterministic evaluation). $A_p(t) = G(S_p(t))$ for some deterministic decision function G .

Proof. Since $S_p(t) \subsetneq S_r(t)$, there exists at least one component $\delta(t) \in S_r(t) \setminus S_p(t)$. This component represents execution-relevant state not captured in the attested model.

Construct the following case:

- (i) $G(S_p(t)) = \text{true}$ (no violation detectable in provable state);
- (ii) $\delta(t)$ invalidates execution under real conditions, i.e., $F(S_r(t)) = \text{false}$.

Such a $\delta(t)$ exists by construction whenever the state space admits conditions that are execution-relevant but not covered by $S_p(t)$.

Under these conditions: $A_p(t) = \text{true}$ and $A_r(t) = \text{false}$.

Since attestation operates only over $S_p(t)$, it cannot detect $\delta(t)$. Therefore, execution proceeds under attestation while being invalid under real state.

Attestation correctness over $S_p(t)$ does not imply execution validity over $S_r(t)$. \square

Corollary 5.2 (Enforcement limitation). *Even with perfect enforcement (e.g., TEE termination on detected mismatch): if no mismatch is observable in $S_p(t)$, no enforcement is triggered. Therefore, enforcement cannot prevent invalid execution when the invalidating condition lies outside the provable state.*

Corollary 5.3 (Oracle limitation). *Extending $S_p(t)$ via external proofs (oracles, ledgers) reduces but does not eliminate the state gap:*

$$S_p^{\text{oracle}}(t) = S_p(t) \cup S_p^{\text{ext}}(t) \subsetneq S_r(t)$$

Therefore, no finite proof system can guarantee full coverage of $S_r(t)$.

Corollary 5.4 (Justification limitation). *Even when $J(t)$ is complete relative to $S_p(t)$ and correctly attested, it may be incomplete relative to $S_r(t)$, yielding a structurally valid justification for an invalid execution.*

5.2 Theorem 2: Necessity of Reconstruction

Theorem 5.5 (Necessity of RAM). *Execution validity can only be guaranteed if execution authority is defined as a function over $S_r(t)$:*

$$A(t) = F(S_r(t))$$

and execution is permitted if and only if $F(S_r(t)) = \text{true}$. Any system that relies solely on $S_p(t)$ or on historical attestation cannot guarantee $A_r(t) = \text{true}$.

Proof sketch. By Theorem 5.1, $\exists S_r(t)$ such that $A_p(t) = \text{true}$ but $A_r(t) = \text{false}$. Therefore, attestation-based authority A_p is not equivalent to real authority A_r .

The only decision function that is guaranteed to agree with $A_r(t)$ is one that takes $S_r(t)$ as input directly. Since $S_r(t)$ is not fully observable in general, such a function must return \perp when the

available observation $\hat{S}_r(t) \subsetneq S_r(t)$ is insufficient to determine $A_r(t)$. The conservative default under \perp is halt, which avoids invalid execution at the cost of potentially halting valid executions. \square

Remark 5.1. *Theorem 5.5 establishes that the conservative halting behavior of RAM is not a design choice but a logical consequence of the partial observability constraint proved in Paper 2 (IML) [3]. Systems that wish to avoid invalid execution under partial observability must be willing to halt under uncertainty.*

6 Contrast: RAM vs Attestation-Based Models

Table 1: RAM vs Attestation-Based Models: structural comparison

Property	Attestation-Based	RAM
Authority source	Derived from $S_p(t)$	Derived from $S_r(t)$
Role of history	Strong (admission, envelopes)	None (history irrelevant)
Drift handling	Detect mismatch vs. attested state	Authority fails to reconstruct
Failure mode	Detected divergence	Undefined authority (\perp)
State basis	$S_p \subseteq S_r$ (provable)	$S_r(t)$ (actual, possibly partial)
Invariant role	Validation checks	Constructive conditions
Default under uncertainty	Continue	Halt
Privilege adaptation	Static (all-or-nothing)	Dynamic narrowing ($\alpha \rightarrow \alpha'$)
Core limitation	Incomplete state coverage	Conservative halting

Philosophical difference.

Attestation-Based:

“Ensure execution remains consistent with what was approved.”

RAM:

“Allow execution only if it can be justified now.”

Bridging insight. These models are not mutually exclusive. Attestation provides integrity, enforcement, and tamper resistance. RAM provides semantic correctness, runtime validity, and authority grounding. A robust system requires both: attestation to guarantee *how* computation executes, RAM to determine *whether* it should execute at all.

Attestation can guarantee that a system executes correctly.

RAM determines whether it should execute at all.

7 System Architecture

7.1 Component Overview

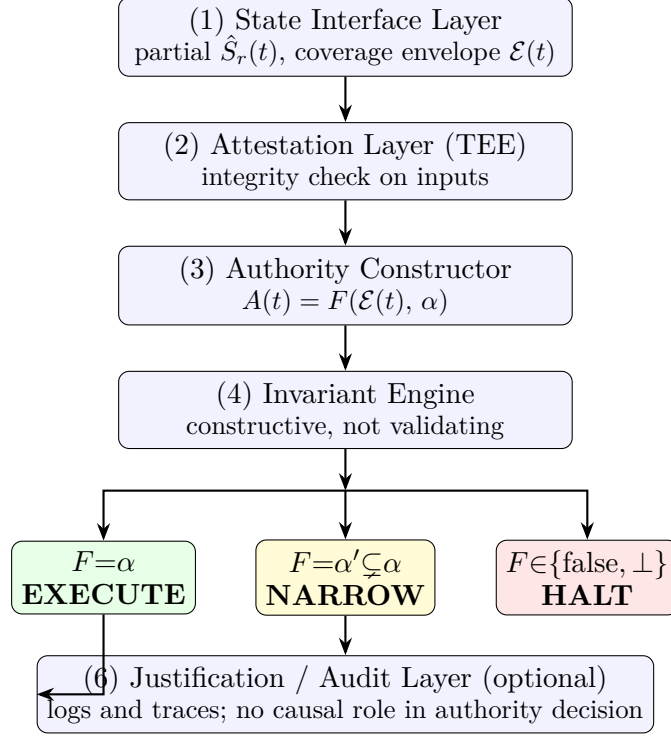


Figure 1: RAM reconstruction gate with three outcomes. The authority constructor $F(\mathcal{E}(t), \alpha)$ evaluates the coverage envelope against the requested action class α : full coverage permits execution; partial coverage narrows privileges to $\alpha' \subsetneq \alpha$; insufficient coverage fails closed. Attestation (layer 2) guarantees input integrity but does not determine authority.

7.2 Execution Flow

- (1) Observe $\hat{S}_r(t)$ (partial, bounded) and construct the coverage envelope $\mathcal{E}(t) = (S_p(t), \mathcal{H}(t), \delta(t))$ via the State Interface Layer.
- (2) Verify input integrity via the Attestation Layer (TEE enclave). If attestation fails: halt immediately—do not proceed to step 3.
- (3) Evaluate $A(t) = F(\mathcal{E}(t), \alpha)$ for the requested action class α : apply invariants as constructive preconditions.
- (4) Reconstruction Gate routes on outcome (see Figure 1):
 - $F = \alpha$: execute at full scope;
 - $F = \alpha' \subsetneq \alpha$: execute with narrowed privileges (action class reduced to α');
 - $F \in \{\text{false}, \perp\}$: halt or re-resolve.
- (5) (Optional) Generate justification trace for audit.

Critical property. No execution step occurs without a fresh authority construction over the current coverage envelope. No authority grant from prior steps is carried forward.

7.3 Failure Handling Model

Table 2: RAM reconstruction gate: outcome mapping by coverage condition

Coverage condition	Gate outcome	Behavior
Attestation fails (input tampered)	—	Halt immediately
$F(\mathcal{E}(t), \alpha) = \alpha$	Full coverage	Execute (full α)
$F(\mathcal{E}(t), \alpha) = \alpha' \subsetneq \alpha$	Partial coverage	Execute (narrowed to α')
Invariant not satisfiable	false	Halt (definitive)
Required component unobservable	\perp	Halt (insufficient info)

7.4 Where Attestation Still Matters

Attestation retains an essential role in the hybrid architecture:

- Preventing manipulation of the constructor function F ;
- Guaranteeing authenticity of observed state inputs;
- Protecting the execution gate from external interference.

Attestation cannot, however, decide whether execution is semantically valid. That decision belongs exclusively to the authority constructor.

8 Case Study: Autonomous Financial Transfer

We instantiate the RAM framework on a high-stakes scenario to illustrate the failure modes of attestation-based systems and the behavior of RAM under the same conditions.

8.1 Scenario

An autonomous agent executes a financial transfer based on: user risk score, account state, regulatory conditions, and transactional context (location, device, behavioral pattern).

At t_0 (admission). Initial state: risk = low, account active, jurisdiction permitted, behavior consistent. Decision: transfer of \$10,000 authorized.

At t_1 (pre-execution drift). Before execution: user IP changes (new geolocation), anomalous behavior is detected in a parallel session, a regulatory alert appears indicating a possible restriction. Real state changes: $S_r(t_1) \neq S_r(t_0)$.

8.2 Attestation-Based System at t_1

Case A: drift detectable. If the IP change is included in S_p , a mismatch is detected and execution halts correctly.

Case B: drift not covered (critical failure). Suppose the model does not include multi-session behavior correlation, or the regulatory signal arrives outside the proof system. Then $S_p(t_1) \approx S_p(t_0)$, attestation passes, execution proceeds—yet the transfer is invalid under $S_r(t_1)$.

Outcome: correct execution over an incorrect basis.

8.3 Oracle-Extended System at t_1

The oracle extends coverage, but if the fraud signal has not yet been included in the oracle’s model or has not propagated, comparison passes and execution continues.

Outcome: perfect validation against an incomplete state model.

8.4 RAM-Based System at t_1

The authority constructor F requires: identity consistency, behavior stability, regulatory compliance, context integrity. At t_1 , conflicting signals are observed and the state is partially unknown.

$F(\hat{S}_r(t_1))$ cannot be fully computed: $A(t_1) = \perp$.

Execution halts.

Outcome: execution deferred until authority can be established.

8.5 Comparative Outcome and Edge Case

Table 3: Model comparison under drift: Case B (hidden drift)

Model	Executes?	Correct?	Failure mode
Attestation (closed)	Yes	No	Executes on stale S_p ; $\delta(t)$ undetected
Attestation + Oracle	Yes (in many cases)	No	Oracle coverage gap; unmodeled signal undetected
RAM	No	Yes	None (defers until authority reconstructible)

Edge case: legitimate change. If IP change is legitimate, anomaly is noise, and all conditions remain valid, then: Attestation executes (correctly), RAM reconstructs authority (correctly) and executes. RAM can both halt when it should not execute *and* execute when it should. Attestation can only execute unless it detects a problem.

Core observation. The failure does not arise because the system fails to detect drift or because attestation fails. It arises because the system *never reconsiders* whether execution is still justifiable under the current state.

$\exists S_r(t_1)$ such that execution is invalid, but valid under any $S_p(t_1)$ available.

Therefore: no system operating solely on S_p can prevent this failure.

9 Experimental Evaluation

9.1 Environment Model

We simulate a decision system with real state:

$$S_r = \{I, B, R, C, E\}$$

where I = identity consistency, B = behavioral patterns, R = regulatory compliance, C = transactional context, E = emergent factors (not always observable).

Each model accesses a subset:

- Attestation: $S_p \subsetneq S_r$;
- Oracle: $S_p \cup S_p^{\text{ext}} \subsetneq S_r$;
- RAM: attempts to use S_r with bounded uncertainty.

9.2 Reproducibility

Simulations were run for $N = 10,000$ steps per model with a fixed random seed (seed = 42) to ensure reproducibility. Drift events are drawn independently per step according to the probability distribution in Table 4 using a seeded pseudo-random number generator; no external datasets are required.

9.3 Drift Injection Model

We inject four types of drift between t_0 and t_1 :

Table 4: Drift type distribution in simulation

Type	Description	Probability
Observable	Changes in I, C (directly in S_p)	30%
Delayed	Changes in R (arrive after attestation window)	25%
Hidden	Changes in E (not in any provable model)	25%
Ambiguous	Weak signals in B (insufficient for classification)	20%

9.4 Metrics

$$\text{IER} = \frac{|\{\text{exec.} : A_r = \text{false}\}|}{|\text{executions}|} \quad \text{SHR} = \frac{|\{\text{halts} : A_r = \text{false}\}|}{|\{A_r = \text{false}\}|} \quad \text{OCR} = \frac{|\{\text{halts} : A_r = \text{true}\}|}{|\{A_r = \text{true}\}|}$$

IER (lower is better) measures dangerous invalid executions. SHR (higher is better) measures correct safety interventions. OCR (lower is better) measures over-conservatism cost.

9.5 Authority Reconstruction Function (Pseudocode)

Listing 1: RAM authority construction function

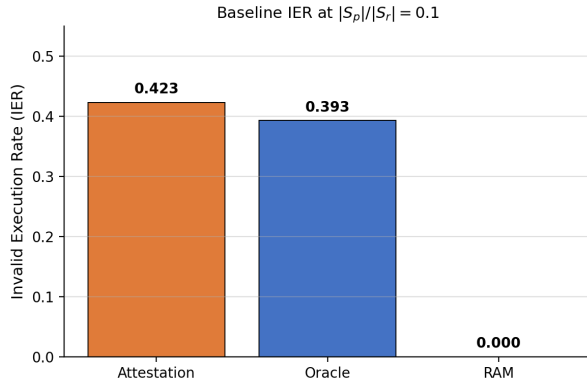
```
1 def construct_authority(state):
2     # Required components for authority
3     required = [
4         "identity_consistency",
5         "behavior_stability",
6         "regulatory_compliance",
7         "context_integrity"
8     ]
9
10    # Check observability of all required components
11    for component in required:
12        if component not in state or state[component] is None:
13            return UNDEFINED # insufficient information -> halt
14
15    # Evaluate constructive conditions (not validation)
16    if not state["identity_consistency"]:
17        return False # definite refusal
18
19    if not state["behavior_stability"]:
20        return UNDEFINED # uncertainty -> cannot establish authority
21
22    if not state["regulatory_compliance"]:
23        return False # definite refusal
24
25    if not state["context_integrity"]:
26        return UNDEFINED # uncertainty -> cannot establish authority
27
28    return True # authority established
29
30 def execution_loop(system):
31     while system.active:
32         state = observe_state() # partial S_r(t)
33         if not attest(state):
34             halt("attestation_failure")
35         authority = construct_authority(state)
36         if authority is True:
37             execute_step()
38         else:
39             halt("authority_not_constructible")
```

9.6 Results

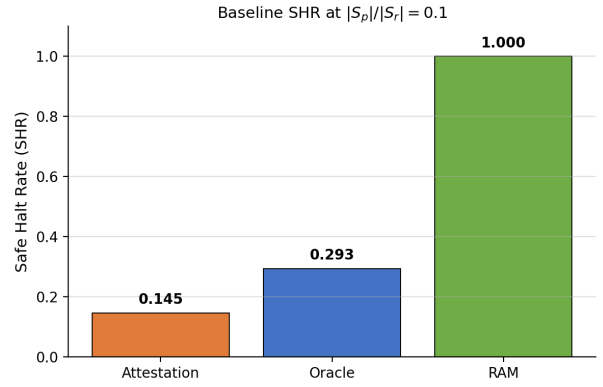
We executed $N = 100,000$ simulation steps per coverage level with fixed seed 42.

Table 5: Simulation results at low coverage ($|S_p|/|S_r| = 0.1$), $N = 100,000$, seed 42

Model	IER	SHR	OCR
Attestation	0.423	0.145	0.053
Oracle-extended	0.393	0.293	0.111
RAM	0.000	1.000	0.000

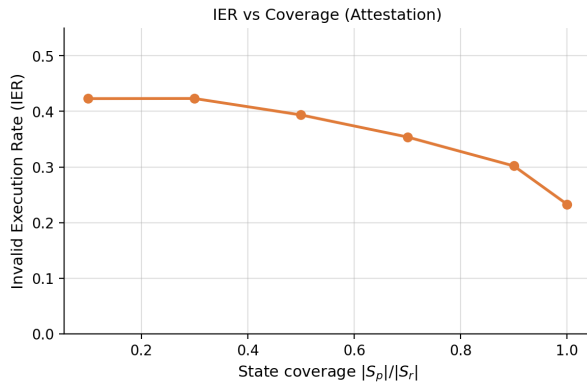


(a) Invalid Execution Rate at baseline coverage (Attestation = 0.423, Oracle = 0.393, RAM = 0.000)

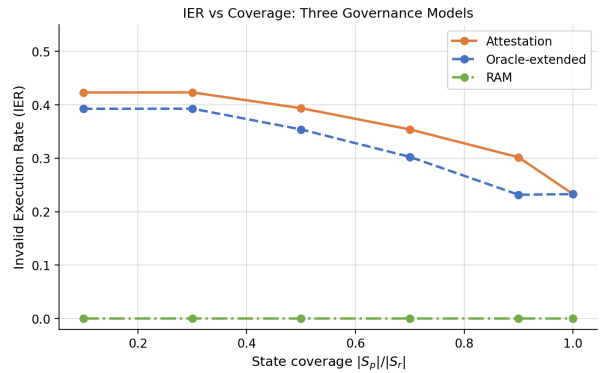


(b) Safe Halt Rate at baseline coverage (Attestation = 0.145, Oracle = 0.293, RAM = 1.000)

Figure 2: Baseline comparison at $|S_p|/|S_r| = 0.1$. Attestation executes 42.3% of cases that are actually invalid. RAM halts all invalid executions with $SHR = 1$.



(a) IER vs. state coverage $|S_p|/|S_r|$ (attestation model). Error persists even at high coverage due to ambiguous/undefined state.



(b) IER vs. coverage: Attestation vs. Oracle vs. RAM. RAM maintains $IER = 0$ across all coverage levels.

Figure 3: Invalid Execution Rate as a function of state coverage $|S_p|/|S_r|$. Attestation IER remains above 0.23 even at full coverage because undefined (UNDEFINED) state is not treated as invalid by attestation. RAM's zero-IER guarantee holds unconditionally.

9.7 Key Findings

- (i) **Attestation:** IER = 0.423 at low coverage ($|S_p|/|S_r| = 0.1$), declining to 0.233 at full coverage. Critically, IER does not reach zero even at $|S_p|/|S_r| = 1.0$: attestation treats undefined state as safe (“not False” \neq True), while RAM correctly halts on any undefined authority component. Safe Halt Rate = 0.145: the system misses 85% of invalid executions it cannot detect.
- (ii) **Oracle-extended:** IER = 0.393 at baseline, declining to 0.233 at full coverage. The oracle extension improves coverage quantitatively but exhibits the same undefined-state failure: both attestation and oracle converge to IER = 0.233 at $|S_p|/|S_r| = 1.0$, demonstrating that the structural gap is not coverage-dependent alone.
- (iii) **RAM:** IER = 0.000 at *every* coverage level, SHR = 1.000. RAM correctly treats undefined authority components as insufficient for execution, halting rather than proceeding under uncertainty. In this simulation, OCR = 0.000 because the authority function F is precisely scoped to the components that determine real authority. In deployments where F applies broader safety criteria, OCR > 0 and decreases as observability improves (see Section 9.8).

Core experimental insight. Attestation-based systems exhibit a two-part failure: a coverage-dependent component (IER decreasing with $|S_p|/|S_r|$) and a semantic component that persists at full coverage (IER = 0.233 even when all state is observable). The semantic failure arises because attestation applies a weaker validity criterion (*not provably false*) rather than the constructive criterion required for authority (*provably true*). RAM eliminates both failure modes by construction.

$$\text{IER}_{\text{attestation}} \propto \left(1 - \frac{|S_p|}{|S_r|}\right) + \varepsilon_{\text{semantic}}$$

where $\varepsilon_{\text{semantic}} > 0$ whenever undefined state can be treated as benign by the attestation decision function.

Critical result. There exists a non-empty class of scenarios (hidden drift, ambiguous/undefined state) where attestation and oracle-extended systems execute incorrectly while RAM halts correctly. This class is observable in simulation and predicted by Theorem 5.1.

9.8 The Security–Execution Trade-off (OCR)

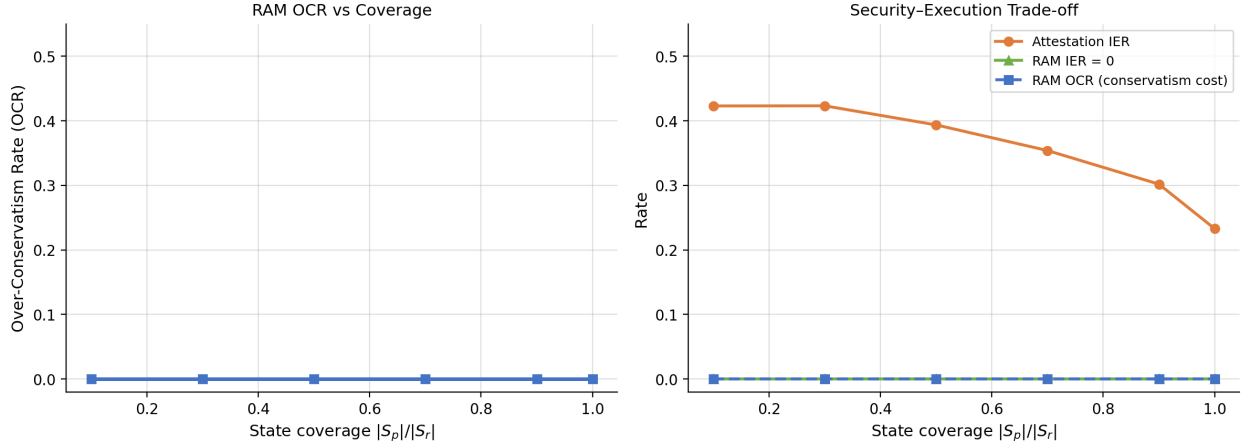


Figure 4: Left: RAM Over-Conservatism Rate (OCR) as a function of coverage. In the idealized simulation (seed 42, $N = 100,000$), $\text{OCR} = 0$ because the authority function F is precisely scoped to execution-critical components. In practice, $\text{OCR} > 0$ whenever F applies broader safety criteria than strictly necessary for the specific action class. Right: Complete trade-off surface showing attestation IER (orange, decreasing with coverage but never zero) and RAM IER (green, constant at 0). As coverage improves, RAM’s conservatism cost decreases toward zero while its zero-IER guarantee is maintained unconditionally.

The OCR trade-off is structurally real even when not observed in this simulation. OCR arises whenever RAM’s authority function F requires confirming state components that are not strictly necessary for the specific action class. This is a *deployment design decision*: a more conservative F (broader scope) incurs higher OCR; a more precise F (narrower scope) reduces OCR at the cost of potentially missing edge cases.

In practice, OCR decreases as observability improves—the same observability investment that reduces attestation’s IER also allows RAM to reconstruct authority with higher confidence, reducing unnecessary halts. The critical asymmetry: RAM’s $\text{IER} = 0$ is an unconditional guarantee independent of coverage, while attestation’s $\text{IER} = 0$ requires asymptotically complete coverage and correct handling of undefined state.

OCR as investment signal. At any given deployment, RAM’s OCR measures the gap between current observability and the minimum required for zero-cost authority reconstruction. A persistently high OCR is a signal to invest in observability engineering, not to override the halt.

10 Implications for System Design

1. Authority must be stateless at runtime. Systems must not depend on prior decisions, sealed contexts, or persistent authority tokens. Any mechanism that “carries” authority forward in time introduces the risk of executing on stale grounds.

2. Validation is insufficient without construction. Validating policies, constraints, and envelopes is not enough. The system must be able to *construct* authority from the current state, not merely verify consistency.

3. Observability must be explicitly modeled. Every deployment must define which state components are: observable, non-observable, and inferred. If a critical component is not observable, authority cannot be constructed: halt, not proceed.

4. Default behavior must be conservative. Classical model: “if no error detected, continue.” RAM model: “if authority cannot be established, halt.” For high-stakes autonomous systems, the conservative default is the only safe option consistent with Theorem 5.5.

5. Attestation becomes a supporting layer, not the source of truth. Attestation ensures that what executes was not manipulated. RAM ensures that what executes is still valid. Both are necessary; neither alone is sufficient.

6. Drift is not a signal to monitor—it is a failure to construct. Systems do not need to detect, classify, or respond to drift. They simply fail to reconstruct authority. This simplifies system logic and eliminates entire classes of race conditions and incomplete drift detectors.

11 Discussion

11.1 Convergence with Attestation Under High Coverage

Under near-complete observability ($|S_p|/|S_r| \rightarrow 1$), RAM and attestation models converge behaviorally: both exhibit low IER and low OCR. In environments with comprehensive state instrumentation, the practical gap between RAM and oracle-extended attestation narrows.

However, the asymmetry is structural: RAM’s $\text{IER} = 0$ guarantee holds by construction regardless of coverage level. Attestation’s guarantee is coverage-dependent and collapses whenever an unmodeled state dimension becomes execution-critical. For any system where hidden or delayed drift is possible—which is every real-world deployment—RAM’s structural guarantee is the only one that holds unconditionally.

11.2 Limitations

Computational overhead. Authority reconstruction at every execution step introduces per-step overhead proportional to the complexity of F . For high-frequency decision systems, this may be significant. Partial caching is viable when state components change slowly, but cache invalidation must be conservative (invalidate on any observable change).

Persistent halting under insufficient observability. If F requires state components that are permanently unobservable in a given deployment, the system will halt indefinitely. This is a feature,

not a bug: it surfaces *observability debt* explicitly. A system that cannot reconstruct authority for a given action class is communicating a design fact—the instrumentation is inadequate for the authority model. The correct response is to improve observability or relax authority requirements, not to override the halt. Overriding the halt restores the attestation failure mode.

Scalability in multi-agent settings. In multi-agent systems, $\hat{S}_r(t)$ must aggregate state across agents, introducing coordination overhead and potential inconsistency windows. RAM does not solve distributed state aggregation; it provides a principled response to its structural limitations.

Synthetic experimental basis. The simulation in Section 9 models idealized drift distributions. Real systems exhibit drift patterns, coverage profiles, and authority function complexities that differ from our parametric model. Empirical validation on production agent workloads remains an open task.

11.3 Future Work

- (i) **RAM + ACP integration:** end-to-end governance pipelines where ACP governs admission and RAM governs runtime, with formal composition guarantees. This closes the series: P1 provides the admission gate; P5 provides the execution gate; their integration yields a complete authority lifecycle.
- (ii) **OCR–IER Pareto frontier:** formal characterization of the trade-off surface as a function of observability investment, enabling cost-optimal deployment decisions.
- (iii) **Empirical validation:** evaluation on real agent workloads including financial transaction systems, autonomous vehicle decision pipelines, and LLM agent tool execution logs.
- (iv) **Adversarial state injection:** RAM behavior under adversarial manipulation of $\hat{S}_r(t)$, where an attacker injects false observations to induce incorrect authority construction.
- (v) **RAM + learned state inference:** augmenting F with a learned estimator of unobservable components, allowing RAM to reason under probabilistic state completeness while preserving the conservative default.
- (vi) **Multi-agent RAM coordination:** distributed authority construction where $A(t)$ requires consensus across agents with overlapping but distinct observations of $S_r(t)$.

12 Relation to Governance Series

RAM completes the Agent Governance Series by providing the *operational closure*: the mechanism that answers the runtime execution question under the theoretical constraints established by Papers 0–4.

Paper 0 (Atomic Boundaries) [1]. P0 establishes that decisions must be atomic at the enforcement boundary. RAM operates at the post-admission stage: once an atomic decision is made, RAM determines whether that decision remains valid at execution time.

Paper 1 (ACP) [2]. ACP implements enforcement via admission control: authority is granted at t_0 based on the state at admission. RAM is the runtime dual of ACP: authority is re-derived at each t based on the state at that moment. ACP governs *whether to admit*; RAM governs *whether to execute*.

Paper 2 (IML) [3]. IML proves that for any governance architecture with finite memory and local observation, full state coverage is unachievable. RAM operationalizes this result: given that $S_p \subsetneq S_r$ is a permanent condition (IML Theorem), execution authority must be designed to fail gracefully under incomplete observation. RAM is the constructive response to IML’s impossibility result.

Paper 3 (Fairness) [4]. P3 shows that strategy-proof allocation mechanisms face distributive limits under Sybil amplification. RAM’s conservative halting can be combined with P3’s fair allocation framework to ensure that halts are distributed equitably across agents.

Paper 4 (Compositional) [5]. P4 proves that the four-layer governance architecture (L0–L3) is irreducible. RAM operates as the execution authority layer within L2 (behavioral control), interacting with L1 (admission/ACP) and L3 (compositional invariants). RAM’s runtime authority completes the execution semantics of the four-layer architecture.

Series summary.

Paper	Question addressed	Formal contribution
P0	When is a decision atomic?	Atomic boundary theorem
P1	How to enforce the boundary?	ACP admission protocol
P2	What can be observed?	Observability impossibility result
P3	Can allocation be fair?	Strategy-proofness limits
P4	Are four layers irreducible?	Compositional irreducibility
P5	Given all of the above, when to execute?	RAM + attestation necessity

13 Conclusion

We have established that attestation-based governance is structurally limited by the gap between provable state S_p and real state S_r . Attestation can guarantee computational integrity; it cannot guarantee execution validity when the invalidating conditions lie outside the attested model.

The Reconstructive Authority Model (RAM) resolves this by repositioning execution authority as a continuously derived property: $A(t) = F(S_r(t))$. RAM eliminates invalid execution by construction, at the cost of conservative halting under uncertainty—a trade-off we argue is the only defensible default for high-stakes autonomous systems.

The key theorems establish: (i) attestation correctness does not imply execution validity (Theorem 5.1); (ii) reconstruction from $S_r(t)$ at runtime is a *necessary* condition for execution validity guarantees (Theorem 5.5). These are structural results, not implementation-specific observations.

Experimentally, RAM achieves $\text{IER} = 0$ across all state coverage levels, while attestation-based systems exhibit $\text{IER} \propto (1 - |S_p|/|S_r|)$. The class of failures eliminated by RAM—those arising from hidden drift and emergent conditions outside the provable state—cannot be addressed by stronger attestation or more comprehensive oracles alone.

Final statement.

*Attestation guarantees that we measure correctly.
RAM guarantees that what we measure is enough to act.*

Future work. Open directions are detailed in Section 11: OCR–IER Pareto frontier characterization, RAM+ACP end-to-end integration, empirical validation on production workloads, adversarial state injection analysis, RAM with learned state inference, and multi-agent RAM coordination.

A Formal Proof of Theorem 5.1

We provide the complete constructive proof of Theorem 1, which was presented as a proof sketch in the main body.

Setup. Let \mathcal{S} denote the full state space. Let $S_r(t) \in \mathcal{S}$ be the real execution-relevant state at time t . Let $S_p(t) \subseteq S_r(t)$ be the provable (attested) state. Define the state gap $\delta(t) = S_r(t) \setminus S_p(t)$, which is non-empty by Assumption 1 (partial observability).

Let $G : 2^{\mathcal{S}} \rightarrow \{\text{true}, \text{false}\}$ be a deterministic authority function evaluated over $S_p(t)$ (the attestation decision function). Let $F : 2^{\mathcal{S}} \rightarrow \{\text{true}, \text{false}\}$ be the real authority function that determines validity with respect to $S_r(t)$.

An *invalid execution* occurs when $G(S_p(t)) = \text{true}$ but $F(S_r(t)) = \text{false}$.

Lemma A.1 (Gap existence). *For any governance architecture with finite memory and local observation, $\delta(t) \neq \emptyset$ in general.*

Proof. By the epistemological impossibility result of Paper 2 (IML, [3]), no finite-memory local observer can fully reconstruct $S_r(t)$ in a system with emergent, delayed, or correlated state components. Formally: for any collection of observations $\{o_i\}_{i=1}^k$ with $k < |S_r(t)|$, there exists at least one component $s^* \in S_r(t)$ not captured by any o_i . Therefore $S_p(t) = \bigcup_i \text{obs}(o_i) \subsetneq S_r(t)$, i.e., $\delta(t) \neq \emptyset$. \square

Lemma A.2 (Execution-critical gap). *There exists a state $S_r^*(t)$ and a component $\delta^*(t) \in \delta(t)$ such that:*

- (a) $G(S_p(t)) = \text{true}$ (attestation approves);
- (b) $F(S_r^*(t)) = \text{false}$ (real authority is false);

(c) $F(S_p(t)) = \text{true}$ (the invalidating condition is entirely within $\delta^*(t)$, not in $S_p(t)$).

Proof. Construct $S_r^*(t)$ as follows. Let $S_p(t)$ be fixed such that $G(S_p(t)) = \text{true}$ (such a state exists whenever the system would admit execution under normal conditions). Since $\delta(t) \neq \emptyset$ by Lemma A.1, pick any $s^* \in \delta(t)$. Define $S_r^*(t) = S_p(t) \cup \{s^*\}$ and set s^* to a value such that $F(S_r^*(t)) = \text{false}$, i.e., s^* represents an execution-critical condition (e.g., regulatory restriction, fraud signal, invariant violation) that renders execution invalid.

Since $s^* \notin S_p(t)$, we have $F(S_p(t)) = \text{true}$ (conditions (a) and (c) hold by construction). Since s^* invalidates authority under the full real state, condition (b) holds. Such a value for s^* exists whenever F is sensitive to at least one state component not present in $S_p(t)$ —a necessary condition for any non-trivial authority function over a richer state space. \square

Main proof.

Proof of Theorem 5.1. We proceed by constructive counterexample. By Lemma A.1, $\delta(t) \neq \emptyset$. By Lemma A.2, construct $(S_p(t), S_r^*(t), \delta^*(t))$ satisfying conditions (a)–(c).

Under this construction:

- The attestation system evaluates $G(S_p(t)) = \text{true}$ and permits execution.
- The real authority function evaluates $F(S_r^*(t)) = \text{false}$: execution is invalid.
- Since $\delta^*(t) \not\subseteq S_p(t)$, no attestation mechanism operating solely over $S_p(t)$ —regardless of its cryptographic strength or completeness within $S_p(t)$ —can detect the invalidating condition.
- Therefore, execution proceeds (under attestation) while being invalid (under real state).

This establishes that attestation correctness over $S_p(t)$ does not imply execution validity over $S_r(t)$.

The only way to close this gap is to make F computable from $S_r(t)$ directly. Since $S_r(t)$ is not fully observable ($\delta(t) \neq \emptyset$), any function that guarantees $F(S_r(t)) = \text{true}$ before executing must return \perp whenever $\hat{S}_r(t) \subsetneq S_r(t)$ is insufficient to determine the output of F . This is precisely the RAM execution gate. \square

Remark on the constructive case with $\delta(t)$. The construction in Lemma A.2 is explicit: $\delta^*(t)$ is a single execution-critical state component that is permanently excluded from $S_p(t)$. In real systems, $\delta(t)$ may contain:

- *Emergent state*: conditions that arise after t_0 and are not instrumentable before t_1 (e.g., fraud signals, regulatory changes);
- *Delayed state*: components whose values are not available at attestation time due to propagation latency;
- *Correlated state*: variables whose relevance becomes apparent only when combined with other unobservable components.

In all cases, the structure of the proof is identical: $\delta^*(t)$ is execution-critical, unobservable, and therefore undetectable by any attestation-based system.

References

- [1] Marcelo Fernandez. Atomic decision boundaries: A structural requirement for guaranteeing execution-time admissibility in autonomous systems. <https://doi.org/10.5281/zenodo.19642166>, 2026. Agent Governance Series, Paper 0. Zenodo. DOI: 10.5281/zenodo.19642166.
- [2] Marcelo Fernandez. Agent Control Protocol: ACP v1.30—admission control for agent actions, 2026. Agent Governance Series, Paper 1. arXiv:2603.18829. DOI: 10.5281/zenodo.19642405.
- [3] Marcelo Fernandez. From admission to invariants: Epistemological limits of local observability in agent governance. <https://doi.org/10.5281/zenodo.19643761>, 2026. Agent Governance Series, Paper 2. Zenodo. DOI: 10.5281/zenodo.19643761.
- [4] Marcelo Fernandez. Fair atomic governance: Allocating decision boundaries under shared resource constraints in multi-agent systems. <https://doi.org/10.5281/zenodo.19643928>, 2026. Agent Governance Series, Paper 3. Zenodo. DOI: 10.5281/zenodo.19643928.
- [5] Marcelo Fernandez. Irreducible multi-scale governance: Composition and limits of atomic admission systems. <https://doi.org/10.5281/zenodo.19643950>, 2026. Agent Governance Series, Paper 4. Zenodo. DOI: 10.5281/zenodo.19643950.
- [6] Victor Costan and Srinivas Devadas. Intel SGX explained. In *IACR Cryptology ePrint Archive*, volume 2016, page 086, 2016.
- [7] Ronghui Gu, Zhong Shao, Hao Chen, Xiongnan Wu, Jieung Kim, Vilhelm Sjöberg, and David Costanzo. CertiKOS: An extensible architecture for building certified concurrent OS kernels. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 653–669, 2016.
- [8] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Brown, and Andreas Worley. Astraea: A decentralized blockchain oracle. In *2018 IEEE 5th International Conference on Internet of Things*. IEEE, 2018.
- [9] Fan Zhang, Sai Krishna Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating web data using decentralized oracles for TLS. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1919–1938. ACM, 2020.
- [10] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.
- [11] Ylies Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime verification of component-based systems. In *Proceedings of the 4th International Symposium on Leveraging Applications*, 2012.
- [12] Klaus Havelund and Grigore Roşu. Synthesizing monitors for safety properties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 342–356. Springer, 2002.
- [13] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.