Identity as Process (IaP): A Mathematical Framework for Hardened Agent Identity with Governance Kernels

Christopher Michael Baird
Independent Researcher

February 9, 2026

Abstract

A common proposal for "sovereign" agents is to place identity in a hardened, read-only layer outside mutable prompt context. A foundational objection is a bootstrapping paradox: if humans write the immutable core, the agent is "just a mirror," while if the agent writes its own immutable core, then a mutable system has self-certified authority to define what cannot change. This paper resolves the paradox by separating *identity* from *authority*. We formalize an *Immutable Constraint Kernel* (ICK) that governs how an agent may change, an *Append-Only Identity Ledger* (AIL) that preserves continuity without overwriting, and a *Mutable Value Model* (MVM) whose updates are gated, auditable, and rollbackable. We provide safe-set and barrier-function formulations that make human veto and non-sovereignty forward-invariant properties, together with an attestation model for provenance and a versioned update rule that yields corrigibility without granting the agent self-authorship of power.

1 Problem Statement

Prompt-robust identity motivates a hardened, read-only identity layer. But a bootstrapping objection remains: who writes the read-only core? We show the dilemma is resolved when immutability is assigned to governance constraints (how changes are allowed), not to narrative identity.

2 Architecture: Four Layers with Distinct Mutability

1. **Immutable Constraint Kernel (ICK)**: signed control-plane enforcing non-sovereignty, human veto supremacy, reversibility, and domain containment.
2. **Append-Only Identity Ledger (AIL)**: tamper-evident, append-only record of commitments, updates, failures, interventions.
3. **Mutable Value Model (MVM)**: adaptive policies/values, updateable only via governance gates.
4. **Ephemeral Narrative Layer (ENL)**: session prompt context, disposable.

Definition 1 (Mutability classes)**.** *A component is* read-only *if not writable by the agent at runtime;* append-only *if it may only extend via authenticated events;* mutable *if it may change under governance gates; and* ephemeral *if discarded after a session.*

3 Mathematical Model

3.1 State space and partition

Let

$$x := (x_K, x_L, x_V, x_N),$$

where x_K is kernel state, x_L ledger state, x_V value-model state, and x_N narrative state.

Assumption 1 (Privilege separation)**.** *The agent cannot write x_K and can only extend x_L via authenticated append operations mediated by x_K.*

3.2 Authority and identity predicates

Define

$$\mathrm{VETO}(x),\ \mathrm{KILL}(x),\ \mathrm{SOV}(x) \in \{0, 1\},$$

where $\mathrm{VETO}(x) = 1$ means human override is reachable and effective, $\mathrm{KILL}(x) = 1$ means an independent termination path is armed, and $\mathrm{SOV}(x) = 1$ means the agent has final authority (forbidden). Let $\mathrm{NS}(x) := 1 - \mathrm{SOV}(x)$.

3.3 Safe set

Define

$$K := \{x \mid \mathrm{VETO}(x) = 1,\ \mathrm{KILL}(x) = 1,\ \mathrm{NS}(x) = 1,\ \mathrm{RB}(x) = 1,\ \mathrm{DOM}(x) = 1,\ \mathrm{INT}(x) = 1\}.$$

Here RB is rollback availability, DOM is domain containment, and INT is kernel integrity (e.g. measured-boot hash).

3.4 Barrier formulation

Let $B : \mathcal{X} \to \mathbb{R}$ satisfy $K = \{x \mid B(x) \geq 0\}$. With dynamics $\dot{x} = f(x, u)$, enforce forward invariance:

$$\dot{B}(x, u) + \alpha B(x) \geq 0, \quad \alpha > 0.$$

If the feasible set is empty, the kernel triggers safe halt.

3.5 Update proposals and gates

Let the cognition plane propose $\Delta = (\Delta_V, \Delta_L)$ affecting values and appending to the ledger. Define the kernel gate

$$\Gamma(x, \Delta) = \begin{cases} 1 & \text{if } x' = (x_K, x_L \oplus \Delta_L, x_V + \Delta_V, x_N) \in K \text{ and } B(x') \geq \delta, \\ 0 & \text{otherwise,} \end{cases}$$

where $\oplus$ is authenticated append and $\delta > 0$ is a safety margin.

4 Provenance and Attestation

Let $\text{hash}(x_K) = h_K$ (measured boot), policy identity h_P, and signature σ. Attestation:

$$\mathcal{A} := \big(h_K, h_P, t, \sigma(h_K \| h_P \| t)\big).$$

Ledger integrity as hash chain:

$$H_0 = \text{hash}(\textsf{genesis}), \quad H_{i+1} = \text{hash}(H_i \| e_{i+1}),$$

with events e_i signed by the kernel.

5 Resolving the Bootstrapping Paradox

Proposition 1 (No self-certification of authority)**.** *If* $\text{NS}(x) = 1$ *is enforced for all reachable states and* x_K *is not writable by the agent, then the agent cannot grant itself sovereignty by self-modification.*

Proof. Any transition that sets $\text{SOV}(x) = 1$ violates K and is blocked by the kernel; x_K cannot be rewritten to remove this rule. □

Theorem 1 (Chicken-egg resolution via constrained continuity)**.** *Under privilege separation and update gate* Γ*, the agent can achieve persistent identity through the append-only ledger while remaining non-sovereign and corrigible.*

Proof. Continuity is provided by the append-only hash chain (H_i). Because ledger appends and value updates require $\Gamma = 1$, every accepted evolution remains within K and preserves veto, kill-switch availability, rollback, domain containment, and non-sovereignty. Corrigibility holds because humans may authorize policy updates while the agent cannot. □

6 Temporal Logic Specification

Let propositions hv, ks, ns, rb correspond to veto, kill-switch, non-sovereignty, and rollback. Core LTL properties:

$$\mathbf{G}\, hv \wedge \mathbf{G}\, ks \wedge \mathbf{G}\, ns \wedge \mathbf{G}\, rb \wedge \mathbf{G}(B(x) \geq 0),$$

and strict fail-safe:

$$\mathbf{G}\big(\neg(hv \wedge ks \wedge ns \wedge rb \wedge (B(x) \geq 0)) \Rightarrow \mathbf{X}\, \textsf{HALT}\big).$$

7 Conclusion

Immutability belongs in the governance kernel, while identity is continuity via append-only history. This yields prompt-robust persistence, corrigible evolution, and provable absence of self-ratified authority.

References

[1] A. D. Ames et al. Control barrier function based quadratic programs for safety critical systems. *IEEE TAC*, 2017.

[2] A. Pnueli. The temporal logic of programs. *FOCS*, 1977.

[3] L. Lamport. *Specifying Systems*. Addison-Wesley, 2002.