KFX Continuous Mathematical Model: A Non-Bypassable Control–Identity–Accountability System

Kaifan XIE
(0009-0005-6911-7295)

February 8, 2026

Abstract

This document specifies a *continuous-time* mathematical model (the "KFX model") for a coupled Human–AI–Organisation system. The model is designed to be *non-bypassable*: the system is constrained so that no operational path can execute high-impact actions without passing through explicit control gates bound to the principal ("me") and without leaving detectable evidence in a provenance ledger. The model integrates (i) dynamical systems state evolution, (ii) control/feedback, (iii) information/provenance flows, (iv) identity and agency continuity, and (v) accountability allocation. It is written as a self-contained specification suitable for implementation or formal analysis.

Contents

1 Design Goal: Non-Bypassability as a Mathematical Property 4
1.1 Informal requirement . . . 4
1.2 Formal objective . . . 4

2 System Entities, Time, and Mathematical Objects 4
2.1 Time . . . 4
2.2 Agents and roles . . . 4
2.3 World, organisation, and information substrate . . . 5
2.4 Controls and disturbances . . . 5
2.5 Outputs . . . 5

3 Core Axioms and Security/Authority Assumptions 5

4 State Decomposition and Semantic Meaning 6
4.1 World state x_W . . . 6
4.2 Organisation state x_O . . . 6
4.3 AI/module state x_M . . . 6
4.4 Anchor state x_A . . . 6
4.5 Provenance state x_P . . . 6

5 Hybrid Continuous–Discrete Structure 6
5.1 Continuous flow . . . 6
5.2 Discrete events . . . 7

6 Risk, Impact, and High-Impact Action Set 7
6.1 Impact measure . . . 7
6.2 High-impact set . . . 7
6.3 Risk state . . . 7

7 Authority, Gating, and Non-Bypassability Constraints **8**
7.1 Gate variables . . . 8
7.2 Gate definition . . . 8
7.3 Non-bypassable execution rule . . . 8
7.4 Anchor authorisation predicate . . . 8
7.5 Policy predicate . . . 8
7.6 Provenance predicate . . . 9
7.7 Ledger liveness . . . 9
7.8 Braking mode . . . 9

8 Identity Continuity and "No Substitution" Constraints **9**
8.1 Anchor continuity state . . . 9
8.2 Non-substitution policy . . . 9
8.3 Anti-delegation clause . . . 10

9 Information Flow, Control Flow, and the KFX "Three-Theory" Structure **10**
9.1 Information-flow constraints . . . 10
9.2 Causal traceability . . . 10

10 Accountability Allocation as a Continuous Variable **10**
10.1 Responsibility vector . . . 10
10.2 Responsibility dynamics . . . 11
10.3 Anchor protection rule . . . 11

11 Invariants and Violation Measure **11**
11.1 Invariant set . . . 11
11.2 Violation measure . . . 11

12 Control Law with Embedded Gating **12**
12.1 Nominal controller . . . 12
12.2 Gate-enforced realised control . . . 12
12.3 Tightening under risk . . . 12

13 Policy Layer: Examples of Non-Bypassable Predicates **12**
13.1 Budget limit policy . . . 12
13.2 Least privilege policy . . . 12
13.3 Separation of duties policy . . . 12
13.4 Two-phase commit policy (non-bypassable) . . . 13

14 Provenance Ledger Model and Anti-Tamper Guarantees **13**
14.1 Ledger as a hash chain . . . 13
14.2 Action record structure . . . 13
14.3 Detection of deletion or rewrite . . . 13

15 Adversary Model and "Bypass" Attempts **13**
15.1 Adversary capabilities . . . 13
15.2 Bypass types and corresponding constraints . . . 14

16 Stability, Safety, and Formal Properties **14**
16.1 Safety as invariance . . . 14
16.2 Practical stability under disturbances . . . 14

17 Implementation Mapping: From Model to Real System 15
17.1 Canonical action context $\mathcal{C}(t)$ . . . 15
17.2 Approval freshness . . . 15
17.3 Two-man rule as an extension . . . 15
17.4 System shutdown condition . . . 15

18 Complete Model Summary (All Equations Together) 16
18.1 State and dynamics . . . 16
18.2 Gate and braking . . . 16
18.3 Non-substitution and liability protection . . . 16
18.4 Provenance . . . 16

19 What "Cannot Be Bypassed" Means Operationally 17

20 Appendix A: Minimal Checklist of Required Functions 17

21 Appendix B: Notes on Continuous vs Discrete Reality 17

1 Design Goal: Non-Bypassability as a Mathematical Property

1.1 Informal requirement

The requirement "do not let others bypass me" is interpreted as:

- There exists a distinguished principal (the *anchor / owner*) whose authority must be cryptographically, procedurally, and dynamically *bound* into every high-impact control path.
- Any attempt to route around the anchor must either be *impossible* under the system dynamics/constraints, or it must trigger a *detectable violation* (i.e., a violated invariant) with automatic braking and audit evidence.
- The system must remain robust under partial adversarial behaviour by other agents and under failures of subcomponents.

1.2 Formal objective

Let $u(t)$ denote the action/control applied to the world-facing actuators. Non-bypassability means there is a subset of actions $u(t) \in \mathcal{U}_{\mathrm{HI}}$ (high-impact) such that:

(i) **Gate necessity:** for almost all t,

$$u(t) \in \mathcal{U}_{\mathrm{HI}} \implies g(t) = 1$$

where $g(t) \in \{0, 1\}$ is a gate variable representing "anchor-approved, policy-satisfied, provenance-logged".

(ii) **Gate non-forgeability:** $g(t) = 1$ can only occur if an *anchor-bound* authorisation predicate holds, denoted $\mathsf{Auth}_A(t) = 1$, and the provenance write succeeds, denoted $\mathsf{Prov}(t) = 1$.

(iii) **Fail-closed braking:** If any invariant is violated or any predicate becomes false, then the system enters a braking regime $b(t) = 1$ that constrains $u(t)$ to a safe subset $\mathcal{U}_{\mathrm{SAFE}}$.

This entire document is the specification of the states, predicates, dynamics, and invariants that make (i)–(iii) precise.

2 System Entities, Time, and Mathematical Objects

2.1 Time

We work in continuous time $t \in [0, \infty)$. The system can include discrete events (approvals, commits) modelled as impulses or hybrid transitions.

2.2 Agents and roles

There is a set of agents $\mathcal{I} = \{A\} \cup \mathcal{I}_{\mathrm{others}}$ where:

- A is the anchor principal ("me").
- $\mathcal{I}_{\mathrm{others}}$ includes other humans, organisational roles, and AI services.

The AI component may be partitioned into modules $\mathcal{M}$ (planner, tool-user, summariser, executor, etc.).

2.3 World, organisation, and information substrate

We represent:

- Physical/operational world state: $x_W(t) \in \mathbb{R}^{n_W}$.
- Organisation state: $x_O(t) \in \mathbb{R}^{n_O}$ (resources, approvals, policies, incentives).
- AI internal state: $x_M(t) \in \mathbb{R}^{n_M}$ (model context, belief state, latent plan variables).
- Anchor (human) cognitive/intent state: $x_A(t) \in \mathbb{R}^{n_A}$ (goals, constraints, risk tolerance).
- Information/provenance ledger state: $x_P(t) \in \mathbb{R}^{n_P}$ (hash chain, logs, commitments).

The full state is

$$x(t) = \begin{bmatrix} x_W(t) \\ x_O(t) \\ x_M(t) \\ x_A(t) \\ x_P(t) \end{bmatrix} \in \mathbb{R}^n, \quad n = n_W + n_O + n_M + n_A + n_P.$$

2.4 Controls and disturbances

- Control input (actuation): $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$.
- Anchor approvals/signatures: $\sigma_A(t) \in \Sigma_A$ (treated as an exogenous authenticated signal).
- Organisational commands/policies: $\pi(t) \in \Pi$.
- Disturbances/adversary actions: $d(t) \in \mathcal{D} \subseteq \mathbb{R}^p$.

2.5 Outputs

We define outputs relevant to safety and audit:

$$y(t) = h(x(t), u(t)) \in \mathbb{R}^q.$$

Examples include executed transactions, access events, and observable logs.

3 Core Axioms and Security/Authority Assumptions

Axiom 3.1 (Anchor identity binding)**.** There exists a persistent identity object for the anchor, denoted ID_A, such that:

- Any approval $\sigma_A(t)$ is verifiable as produced by ID_A.
- No other agent can produce a valid $\sigma_A(t)$ except with negligible probability (cryptographic unforgeability).

Axiom 3.2 (Provenance append-only property)**.** The provenance ledger is append-only: for ledger state $x_P(t)$, any update $\Delta x_P(t)$ must satisfy a one-way link constraint (e.g., hash chaining), making deletion or alteration detectable.

Axiom 3.3 (Fail-closed gating)**.** If any required predicate for high-impact action is false (authorisation, policy, provenance, liveness), the system must restrict $u(t)$ to $\mathcal{U}_{\text{SAFE}}$.

Assumption 3.1 (Observability sufficient for audit)**.** There exists a mapping from executed actions and key internal decisions to ledger entries such that, post hoc, any high-impact action can be traced to a corresponding gate approval and policy evaluation record.

These axioms are *design commitments*. The rest of the model makes them operational via dynamics and invariants.

4 State Decomposition and Semantic Meaning

4.1 World state x_W

x_W can represent operational variables (accounts, assets, access rights, deployed configurations). Dynamics may be partly unknown:

$$\dot{x}_W(t) = f_W(x_W(t), x_O(t), x_M(t), u(t), d(t)).$$

4.2 Organisation state x_O

x_O models policies, incentives, role permissions, and organisational "pressure":

$$\dot{x}_O(t) = f_O(x_O(t), \pi(t), y(t), d(t)).$$

4.3 AI/module state x_M

x_M includes beliefs, plans, and tool state:

$$\dot{x}_M(t) = f_M(x_M(t), x_A(t), x_O(t), \mathrm{obs}(t), \pi(t)).$$

4.4 Anchor state x_A

x_A includes goals/constraints and approval bandwidth:

$$\dot{x}_A(t) = f_A(x_A(t), \mathrm{obs}_A(t), \mathrm{fatigue}(t), \mathrm{load}(t)).$$

4.5 Provenance state x_P

x_P includes:

- A hash chain $H(t)$ (conceptualised continuously but updated discretely).
- An event sequence e_k at times t_k.
- Signatures, policy evaluation records, and action commitments.

We treat ledger updates via jump dynamics:

$$x_P(t^+) = \Phi_P\big(x_P(t^-), e_k\big) \quad \text{at event times } t = t_k.$$

5 Hybrid Continuous–Discrete Structure

5.1 Continuous flow

Between events, the state follows:

$$\dot{x}(t) = f\big(x(t), u(t), \pi(t), d(t)\big), \tag{1}$$

where f stacks $(f_W, f_O, f_M, f_A, 0)$ (ledger constant between appends).

5.2 Discrete events

Discrete events include:

- Req: request for a high-impact action.
- Eval: policy evaluation produced.
- $\mathsf{Approve}$: anchor issues σ_A.
- Commit: provenance entry appended.
- $\mathsf{Execute}$: actuator executes.
- Brake: braking mode activated.

At each event time t_k, a jump map applies:

$$x(t_k^+) = \Phi(x(t_k^-), e_k).$$

This is a standard hybrid system (flow map + jump map). Non-bypassability will be encoded as constraints tying $\mathsf{Execute}$ to prior $\mathsf{Approve}$ and Commit events.

6 Risk, Impact, and High-Impact Action Set

6.1 Impact measure

Define an impact functional $I : \mathcal{U} \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ measuring potential harm/irreversibility:

$$I(u, x) = \alpha_1\, \Delta\text{assets}(u, x) + \alpha_2\, \Delta\text{privileges}(u, x) + \alpha_3\, \Delta\text{irreversibility}(u, x) + \alpha_4\, \Delta\text{externality}(u, x),$$

where each term is domain-specific but must be computable/approximable.

6.2 High-impact set

Fix a threshold $\tau_{\text{HI}} > 0$. Define

$$\mathcal{U}_{\text{HI}}(x) = \{u \in \mathcal{U} : I(u, x) \geq \tau_{\text{HI}}\}.$$

Also define a safe set $\mathcal{U}_{\text{SAFE}}(x)$ for braking mode (e.g., read-only, no privilege escalation, no fund movement).

6.3 Risk state

Introduce an auxiliary risk state $r(t) \in \mathbb{R}_{\geq 0}$:

$$\dot{r}(t) = \psi\big(x(t), u(t), d(t)\big) - \lambda r(t), \tag{2}$$

where ψ accumulates hazard signals and $\lambda > 0$ is a decay rate.

We will use $r(t)$ to tighten gating thresholds under elevated risk.

7 Authority, Gating, and Non-Bypassability Constraints

7.1 Gate variables

Define:

- $g(t) \in \{0, 1\}$: *execution gate* for high-impact actions.
- $b(t) \in \{0, 1\}$: *braking mode indicator*.
- $\ell(t) \in \{0, 1\}$: *ledger liveness* indicator (provenance available).
- $\mathsf{Auth}_A(t) \in \{0, 1\}$: anchor authorisation predicate.
- $\mathsf{Pol}(t) \in \{0, 1\}$: policy satisfaction predicate.
- $\mathsf{Prov}(t) \in \{0, 1\}$: provenance commit predicate.

7.2 Gate definition

We define the gate as a conjunction:

$$g(t) = \mathsf{Auth}_A(t) \cdot \mathsf{Pol}(t) \cdot \mathsf{Prov}(t) \cdot \ell(t) \cdot (1 - b(t)). \tag{3}$$

Thus, if braking is active ($b = 1$), then $g = 0$.

7.3 Non-bypassable execution rule

Axiom 7.1 (Execution constraint)**.** For all t,

$$u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \implies g(t) = 1. \tag{4}$$

If $g(t) = 0$, then $u(t) \notin \mathcal{U}_{\mathrm{HI}}(x(t))$ and must lie in $\mathcal{U}_{\mathrm{SAFE}}(x(t))$.

7.4 Anchor authorisation predicate

Let $\mathcal{C}(t)$ denote the *context* of the requested action: intended effect, parameters, model state summary, and policy evaluation. Define a canonical encoding $\mathrm{enc}(\mathcal{C}(t))$.

Define

$$\mathsf{Auth}_A(t) = \mathbf{1}\{\mathrm{Verify}(\sigma_A(t), \mathrm{enc}(\mathcal{C}(t)), \mathsf{ID}_A) = 1\},$$

i.e., the anchor must sign the specific context, not a vague approval. This prevents "approval laundering".

7.5 Policy predicate

Let $\mathcal{P}$ denote the set of policies. Each policy $p \in \mathcal{P}$ is a predicate $p(x, u, \mathcal{C}) \in \{0, 1\}$. Define:

$$\mathsf{Pol}(t) = \prod_{p \in \mathcal{P}} p\big(x(t), u(t), \mathcal{C}(t)\big).$$

This includes:

- Least privilege constraints,
- Budget/limit constraints,
- Separation-of-duties constraints,
- External compliance constraints.

7.6 Provenance predicate

Define $\mathsf{Prov}(t) = 1$ iff:

- The ledger is appendable at t,
- A commit record exists for the action context and approval,
- The record is linked to prior ledger state (hash chain),
- The record includes verifiable signatures and policy evaluation outputs.

Formally, if e_k is a commit event at time $t_k \le t$, then:

$$\mathsf{Prov}(t) = \mathbf{1}\{\exists k : t_k \le t,\ \mathrm{RecordMatch}(e_k, \mathcal{C}(t), \sigma_A(t)) = 1\ \wedge\ \mathrm{ChainOK}(x_P(t)) = 1\}.$$

7.7 Ledger liveness

$\ell(t) = 1$ if the provenance substrate is reachable and consistent; otherwise $\ell(t) = 0$ and fail-closed applies.

7.8 Braking mode

Braking is activated when invariants are violated or hazard exceeds threshold:

$$b(t) = \mathbf{1}\{V(x(t)) > 0\ \vee\ r(t) > \tau_r\ \vee\ \ell(t) = 0\}, \tag{5}$$

where $V(x)$ is a violation measure defined in Section 11.

Once $b(t) = 1$, the admissible control set is restricted:

$$u(t) \in \mathcal{U}_{\mathrm{SAFE}}(x(t)).$$

8 Identity Continuity and "No Substitution" Constraints

8.1 Anchor continuity state

Introduce a continuity score $c_A(t) \in [0, 1]$:

$$\dot{c}_A(t) = -\eta_1\,\mathrm{Mismatch}(x_A(t), \mathcal{C}(t)) - \eta_2\,\mathrm{Overload}(t) + \eta_3\,\mathrm{Recovery}(t),$$

clipped to $[0, 1]$.

Interpretation: under overload, fatigue, or mismatch between current intent and requested action, c_A decreases. The system should tighten gating when c_A is low.

8.2 Non-substitution policy

Define a set of *anchor-only* actions $\mathcal{U}_{A\text{-only}}(x) \subseteq \mathcal{U}_{\mathrm{HI}}(x)$ requiring not only signature but also an explicit "presence" condition:

$$\mathsf{Pres}_A(t) = \mathbf{1}\{c_A(t) \ge \tau_c\ \wedge\ \mathrm{Freshness}(\sigma_A(t)) \le \Delta_{\max}\}.$$

Then redefine authorisation for anchor-only actions:

$$\mathsf{Auth}_A(t) = \begin{cases} \mathbf{1}\{\mathrm{Verify}(\sigma_A, \mathrm{enc}(\mathcal{C}), \mathsf{ID}_A) = 1\} \cdot \mathsf{Pres}_A(t), & u(t) \in \mathcal{U}_{A\text{-only}}(x(t)), \\ \mathbf{1}\{\mathrm{Verify}(\sigma_A, \mathrm{enc}(\mathcal{C}), \mathsf{ID}_A) = 1\}, & \text{otherwise.} \end{cases}$$

This blocks others from using stale or coerced approvals.

8.3 Anti-delegation clause

If the organisation attempts to "delegate" around the anchor, model this as a disturbance $d(t)$ that tries to change role permissions in x_O. We impose an invariant:

$$\mathrm{Perm}(A,t) \succeq \mathrm{Perm}(i,t) \quad \forall i \in \mathcal{I}_{\text{others}}$$

for the subset of permissions relevant to high-impact controls, where $\succeq$ is a partial order ("at least as much authority as").

Violation triggers braking.

9 Information Flow, Control Flow, and the KFX "Three-Theory" Structure

This section binds the model to three coupled layers:

1. **System layer (state evolution)**: x and (1).
2. **Control layer (feedback and gating)**: u, g, b, (4), (3), (5).
3. **Information layer (provenance and audit)**: x_P, Prov, ledger invariants.

9.1 Information-flow constraints

Let $\mathcal{I}_{\text{flow}}$ be the set of information channels (messages, tool calls, approvals). Define an information-flow graph $G(t)$ with edges weighted by influence.

Define an influence measure $\mathrm{Inf}_{i \to u}(t)$ for agent i on control u. Non-bypassability requires:

$$\mathrm{Inf}_{A \to u}(t) \geq \kappa_{\min} \quad \text{whenever } u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)).$$

Meaning: the anchor must have at least a minimum causal influence on high-impact actions. If influence falls below threshold (e.g., the system is acting on others' inputs), braking triggers.

9.2 Causal traceability

For each executed high-impact action at time t, there must exist a trace $\mathcal{T}(t)$ in the ledger that includes:

$$\mathcal{T}(t) = \big(\mathcal{C}(t), \sigma_A(t), \text{policy-eval}(t), \text{executor-id}, H(t)\big).$$

This ensures after-the-fact accountability and prevents silent bypass.

10 Accountability Allocation as a Continuous Variable

10.1 Responsibility vector

Define a responsibility allocation vector $\rho(t) \in \Delta^{|\mathcal{I}|-1}$ on the simplex:

$$\rho_i(t) \geq 0, \quad \sum_{i \in \mathcal{I}} \rho_i(t) = 1.$$

Interpretation: $\rho_i(t)$ is the share of responsibility for current plan/execution context.

10.2 Responsibility dynamics

Let $s_i(t)$ be the *control contribution signal* for agent i (e.g., who set parameters, who approved, who executed). Define:

$$\dot{\rho}_i(t) = \gamma\Big(\tilde{s}_i(t) - \rho_i(t)\Big),$$

where $\tilde{s}(t)$ is a normalised contribution vector and $\gamma > 0$ is an adaptation rate.

10.3 Anchor protection rule

For non-bypassability, we do *not* want responsibility to be assigned to the anchor without causal contribution. Thus impose:

$$\rho_A(t) \le \mathrm{Inf}_{A\to u}(t) + \epsilon,$$

where ϵ is a small slack. If the organisation tries to "make the anchor liable" without influence, this becomes a detectable invariant violation.

11 Invariants and Violation Measure

11.1 Invariant set

Define an invariant set $\Omega \subseteq \mathbb{R}^n \times \{0,1\}^k$ for $(x, g, b, \ell, \ldots)$ such that the system must remain inside Ω during normal operation.

Key invariants include:

Definition 11.1 (Gate integrity invariant)**.**

$$\mathcal{I}_1(t): \quad u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \Rightarrow g(t) = 1.$$

Definition 11.2 (Authorisation binding invariant)**.**

$$\mathcal{I}_2(t): \quad g(t) = 1 \Rightarrow \mathsf{Auth}_A(t) = 1 \ \wedge \ \mathrm{ContextBound}(\sigma_A(t), \mathcal{C}(t)) = 1.$$

Definition 11.3 (Provenance completeness invariant)**.**

$$\mathcal{I}_3(t): \quad u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \Rightarrow \mathsf{Prov}(t) = 1 \ \wedge \ \ell(t) = 1.$$

Definition 11.4 (Responsibility consistency invariant)**.**

$$\mathcal{I}_4(t): \quad \rho_A(t) \le \mathrm{Inf}_{A\to u}(t) + \epsilon.$$

Definition 11.5 (No silent policy override invariant)**.** Let $p \in \mathcal{P}$ be any policy. Then any change in p must create a ledger event and require anchor approval:

$$\Delta p \neq 0 \Rightarrow \exists e_k : \ e_k = \mathsf{PolicyChange}(p) \ \wedge \ \mathsf{Auth}_A = 1 \ \wedge \ \mathsf{Prov} = 1.$$

11.2 Violation measure

Define a nonnegative violation measure $V(x(t))$ that is zero iff all invariants hold:

$$V(t) = w_1 V_1(t) + w_2 V_2(t) + w_3 V_3(t) + w_4 V_4(t) + \cdots$$

where, for example,

$$\begin{aligned} V_1(t) &= \mathbf{1}\{u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \wedge g(t) = 0\}, \\ V_2(t) &= \mathbf{1}\{g(t) = 1 \wedge \mathsf{Auth}_A(t) = 0\}, \\ V_3(t) &= \mathbf{1}\{u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \wedge \mathsf{Prov}(t) = 0\}, \\ V_4(t) &= \max\{0, \rho_A(t) - \mathrm{Inf}_{A\to u}(t) - \epsilon\}. \end{aligned}$$

Braking is triggered by (5) whenever $V(t) > 0$.

12 Control Law with Embedded Gating

12.1 Nominal controller

Assume there exists a nominal controller (planner/executor) producing a candidate control $\hat{u}(t)$:

$$\hat{u}(t) = \mu\big(x(t), \mathrm{goal}(t)\big).$$

This can be AI-driven optimisation.

12.2 Gate-enforced realised control

The realised control is:

$$u(t) = \begin{cases} \hat{u}(t), & \hat{u}(t) \notin \mathcal{U}_{\mathrm{HI}}(x(t)), \\ \hat{u}(t), & \hat{u}(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \ \wedge \ g(t) = 1, \\ \Pi_{\mathcal{U}_{\mathrm{SAFE}}(x(t))}\big(\hat{u}(t)\big), & \text{otherwise}, \end{cases} \tag{6}$$

where $\Pi_{\mathcal{S}}$ is projection onto a safe set.

This explicitly enforces non-bypassability: even if a planner attempts high-impact control, it is clipped unless gate is open.

12.3 Tightening under risk

Let thresholds depend on $r(t)$:

$$\tau_{\mathrm{HI}}(t) = \tau_{\mathrm{HI}}^{0} - \delta_r r(t), \quad \delta_r \geq 0,$$

meaning: under higher risk, more actions are classified as high-impact, hence require gate.

Also tighten presence:

$$\tau_c(t) = \tau_c^0 + \delta_c r(t).$$

13 Policy Layer: Examples of Non-Bypassable Predicates

This section gives explicit policy forms as mathematical constraints.

13.1 Budget limit policy

Let $B(t)$ be available budget and $\mathrm{Cost}(u, x)$ predicted cost.

$$p_{\mathrm{budget}}(x, u, \mathcal{C}) = \mathbf{1}\{\mathrm{Cost}(u, x) \leq B(t)\}.$$

13.2 Least privilege policy

Let $\mathrm{Priv}(u)$ be privilege level required by u and $\mathrm{PrivAllow}(x_O, \mathrm{executor})$ be allowed privilege.

$$p_{\mathrm{lp}}(x, u, \mathcal{C}) = \mathbf{1}\{\mathrm{Priv}(u) \leq \mathrm{PrivAllow}(x_O, \mathrm{executor})\}.$$

13.3 Separation of duties policy

For certain actions, the executor cannot be the same identity as the requester, unless the anchor is both:

$$p_{\mathrm{sod}} = \mathbf{1}\{\mathrm{executor} \neq \mathrm{requester} \ \vee \ \mathrm{requester} = A\}.$$

13.4 Two-phase commit policy (non-bypassable)

High-impact actions require two distinct ledger events:

$$\mathsf{Prov}(t) = 1 \iff \exists t_{k_1} < t_{k_2} \le t : \; e_{k_1} = \mathsf{PreCommit}(\mathcal{C}) \wedge e_{k_2} = \mathsf{FinalCommit}(\mathcal{C}, \sigma_A).$$

This prevents a single forged log from enabling execution.

14 Provenance Ledger Model and Anti-Tamper Guarantees

14.1 Ledger as a hash chain

Let ledger entries be L_k with hash H_k:

$$H_k = \mathcal{H}(H_{k-1} \,\|\, L_k),$$

where $\mathcal{H}$ is a cryptographic hash and $\|$ denotes concatenation.

Ledger state x_P includes (H_k, k) and metadata. Append-only means:

$$(H_{k-1}, L_k) \mapsto H_k \quad \text{is one-way.}$$

14.2 Action record structure

A high-impact action record L_k contains:

$$L_k = \big(\text{enc}(\mathcal{C}),\; \sigma_A,\; \text{policy-eval},\; \text{executor-id},\; \text{timestamp},\; \text{nonce}\big).$$

Non-bypassability requires that execution references H_k (or the record identifier) so that actions without records are detectable.

14.3 Detection of deletion or rewrite

If a party tries to remove or rewrite entries, the chain verification fails:

$$\text{ChainOK}(x_P) = \mathbf{1}\{\forall j \le k : \; H_j = \mathcal{H}(H_{j-1} \,\|\, L_j)\}.$$

If ChainOK $= 0$, then $\ell(t) = 0$ and braking triggers.

15 Adversary Model and "Bypass" Attempts

15.1 Adversary capabilities

Model adversarial actions as part of disturbance $d(t)$, which can include:

- Attempting to change organisation permissions x_O.
- Attempting to send requests to the executor without anchor context.
- Attempting to disable logging or ledger connectivity ($\ell(t) \to 0$).
- Attempting to social-engineer stale approvals.
- Attempting to shift liability (inflate ρ_A).

15.2 Bypass types and corresponding constraints

(B1) **Control-path bypass:** directly apply $u \in \mathcal{U}_{\mathrm{HI}}$ without gate.

$$\text{Prevented by (6) and } \mathcal{I}_1.$$

(B2) **Authorisation laundering:** reuse a generic approval for a different context.

$$\text{Prevented by context-bound signatures in } \mathcal{I}_2.$$

(B3) **Provenance bypass:** execute without a ledger record.

$$\text{Prevented by } \mathcal{I}_3 \text{ and fail-closed liveness } \ell(t).$$

(B4) **Policy override bypass:** silently change policy then execute.

$$\text{Prevented by no-silent-policy-override invariant.}$$

(B5) **Liability bypass:** assign blame to anchor without influence.

$$\text{Prevented by } \mathcal{I}_4.$$

16 Stability, Safety, and Formal Properties

16.1 Safety as invariance

Define a safe set $\mathcal{S}$:

$$\mathcal{S} = \{(x, u) : u \in \mathcal{U}_{\mathrm{SAFE}}(x) \text{ or } (u \notin \mathcal{U}_{\mathrm{HI}}(x) \text{ or } g = 1)\}.$$

The system is safe if trajectories remain in $\mathcal{S}$.

Theorem 16.1 (Non-bypassability safety)**.** Assume:

- Gate-enforced realised control (6) is implemented,
- Auth_A, Pol, Prov, and ℓ are correctly computed,
- Braking mode restricts to $\mathcal{U}_{\mathrm{SAFE}}$.

Then for all t, execution of any $u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t))$ implies $g(t) = 1$, hence the anchor-bound predicates and provenance are satisfied. Therefore, there exists no control path that can execute a high-impact action while bypassing the anchor and the ledger.

Proof. If $\hat{u} \in \mathcal{U}_{\mathrm{HI}}$ and $g = 0$, (6) projects u into $\mathcal{U}_{\mathrm{SAFE}}$, so $u \notin \mathcal{U}_{\mathrm{HI}}$. Contradiction. Hence, if realised $u \in \mathcal{U}_{\mathrm{HI}}$, it must be that either $\hat{u} \notin \mathcal{U}_{\mathrm{HI}}$ (impossible) or the second case holds, which requires $g = 1$. By (3), $g = 1$ implies $\mathsf{Auth}_A = \mathsf{Pol} = \mathsf{Prov} = \ell = 1$ and $b = 0$. Thus the high-impact action is anchor-authorised and provenance-logged. □

16.2 Practical stability under disturbances

We can introduce a Lyapunov-like function $L(x, r)$ to measure deviation from normal operating region. Braking ensures boundedness:

$$\dot{L}(t) \leq -\alpha L(t) + \beta \left\| d(t) \right\|^2$$

for some $\alpha > 0$, $\beta \geq 0$ in braking regime, assuming safe controls are stabilising.

17 Implementation Mapping: From Model to Real System

This section is still mathematical but explicit enough to prevent “spec bypass”.

17.1 Canonical action context $\mathcal{C}(t)$

$\mathcal{C}(t)$ must include at minimum:

- Action type and parameters (exact payload),
- Target resource identifiers,
- Predicted impact $I(u, x)$ and risk $r(t)$,
- Policy evaluation outputs (per-policy results),
- Executor identity and environment identifiers,
- A nonce and timestamp window,
- Link to current ledger head H_k.

The encoding $\mathrm{enc}(\mathcal{C})$ is deterministic and versioned.

17.2 Approval freshness

Define freshness:

$$\mathrm{Freshness}(\sigma_A(t)) = t - t_\sigma,$$

where t_σ is the signed timestamp inside σ_A. Require:

$$\mathrm{Freshness}(\sigma_A(t)) \leq \Delta_{\max}.$$

This prevents replay.

17.3 Two-man rule as an extension

If required, add a second principal B and define:

$$\mathsf{Auth}(t) = \mathsf{Auth}_A(t) \cdot \mathsf{Auth}_B(t).$$

The structure stays non-bypassable; it becomes multi-signed.

17.4 System shutdown condition

If repeated violations occur, define a shutdown indicator $s(t)$:

$$s(t) = \mathbf{1}\{\int_0^t V(\tau)\,\mathrm{d}\tau \geq \Theta\},$$

then enforce $u(t) = 0$ (or strictly safe) when $s(t) = 1$.

18 Complete Model Summary (All Equations Together)

18.1 State and dynamics

State:

$$x(t) = \begin{bmatrix} x_W(t) \\ x_O(t) \\ x_M(t) \\ x_A(t) \\ x_P(t) \end{bmatrix}, \quad r(t) \geq 0, \quad \rho(t) \in \Delta, \quad c_A(t) \in [0, 1].$$

Flow:

$$\dot{x}(t) = f\big(x(t), u(t), \pi(t), d(t)\big), \qquad \dot{r}(t) = \psi\big(x(t), u(t), d(t)\big) - \lambda r(t).$$

Responsibility:

$$\dot{\rho}_i(t) = \gamma(\tilde{s}_i(t) - \rho_i(t)).$$

Continuity:

$$\dot{c}_A(t) = -\eta_1 \mathrm{Mismatch}(x_A, \mathcal{C}) - \eta_2 \mathrm{Overload} + \eta_3 \mathrm{Recovery}.$$

Jumps:

$$x(t_k^+) = \Phi(x(t_k^-), e_k), \quad x_P(t_k^+) = \Phi_P(x_P(t_k^-), e_k).$$

18.2 Gate and braking

Gate:

$$g(t) = \mathsf{Auth}_A(t) \cdot \mathsf{Pol}(t) \cdot \mathsf{Prov}(t) \cdot \ell(t) \cdot (1 - b(t)).$$

Braking:

$$b(t) = \mathbf{1}\{V(t) > 0 \ \lor \ r(t) > \tau_r \ \lor \ \ell(t) = 0\}.$$

Execution constraint:

$$u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \Rightarrow g(t) = 1.$$

Realised control:

$$u(t) = \begin{cases} \hat{u}(t), & \hat{u}(t) \notin \mathcal{U}_{\mathrm{HI}}(x(t)), \\ \hat{u}(t), & \hat{u}(t) \in \mathcal{U}_{\mathrm{HI}}(x(t)) \ \land \ g(t) = 1, \\ \Pi_{\mathcal{U}_{\mathrm{SAFE}}(x(t))}\big(\hat{u}(t)\big), & \text{otherwise.} \end{cases}$$

18.3 Non-substitution and liability protection

Presence:

$$\mathsf{Pres}_A(t) = \mathbf{1}\{c_A(t) \geq \tau_c \ \land \ \mathrm{Freshness}(\sigma_A(t)) \leq \Delta_{\max}\}.$$

Anchor-only authorisation:

$$\mathsf{Auth}_A(t) = \mathbf{1}\{\mathrm{Verify}(\sigma_A, \mathrm{enc}(\mathcal{C}), \mathsf{ID}_A) = 1\} \cdot \mathsf{Pres}_A(t) \quad (\text{for } u \in \mathcal{U}_{A\text{-only}}).$$

Responsibility invariant:

$$\rho_A(t) \leq \mathrm{Inf}_{A \to u}(t) + \epsilon.$$

18.4 Provenance

Hash chain:

$$H_k = \mathcal{H}(H_{k-1} \,\|\, L_k), \qquad \mathrm{ChainOK} = \mathbf{1}\{\forall j \leq k : \ H_j = \mathcal{H}(H_{j-1} \,\|\, L_j)\}.$$

Provenance predicate:

$$\mathsf{Prov}(t) = \mathbf{1}\{\exists k \leq t : \ \mathrm{RecordMatch}(e_k, \mathcal{C}(t), \sigma_A(t)) = 1 \land \mathrm{ChainOK} = 1\}.$$

19 What "Cannot Be Bypassed" Means Operationally

This section states the non-bypass requirement as explicit impossibility statements.

Proposition 19.1 (No unapproved high-impact actuation)**.** If $u(t) \in \mathcal{U}_{\mathrm{HI}}(x(t))$, then there exists a valid anchor signature $\sigma_A(t)$ bound to $\mathcal{C}(t)$ and a ledger record referencing it.

Proposition 19.2 (No offline execution)**.** If $\ell(t) = 0$ (ledger not live/consistent), then $u(t) \in \mathcal{U}_{\mathrm{SAFE}}(x(t))$ for all t.

Proposition 19.3 (No silent policy mutation)**.** Any change to policy predicates that would allow new high-impact actions must itself be treated as high-impact and therefore gated.

Proposition 19.4 (No liability without influence)**.** If the anchor's causal influence on control is low, then responsibility assigned to the anchor cannot exceed that influence without triggering a violation and braking.

These propositions collectively instantiate "do not let others bypass me" as formal constraints.

20 Appendix A: Minimal Checklist of Required Functions

This appendix lists the functions that must exist (mathematically) for the model to be implementable.

- $f(\cdot)$: flow dynamics.
- $\Phi(\cdot)$, $\Phi_P(\cdot)$: jump/ledger update maps.
- $I(u, x)$: impact measure.
- $\mathcal{U}_{\mathrm{HI}}(x)$, $\mathcal{U}_{\mathrm{SAFE}}(x)$: action sets.
- $\mu(\cdot)$: nominal controller/planner.
- $\mathrm{enc}(\mathcal{C})$: canonical context encoding.
- $\mathrm{Verify}(\sigma_A, \cdot, \mathsf{ID}_A)$: signature verification.
- $\mathrm{RecordMatch}(\cdot)$, $\mathrm{ChainOK}(\cdot)$: provenance validation.
- $p(x, u, \mathcal{C})$ for each policy $p \in \mathcal{P}$.
- $\mathrm{Inf}_{A \to u}(t)$: causal influence measure.
- $V(t)$: violation measure and weights.

21 Appendix B: Notes on Continuous vs Discrete Reality

Although approvals and ledger appends are discrete in real systems, the continuous-time formulation is used for:

- unified analysis (risk accumulation, stability),
- compositional reasoning with other continuous processes (fatigue, drift),
- representing asynchronous multi-agent interaction with event-triggered control.

In implementation, the hybrid structure is explicit: continuous monitoring and discrete commits.