

# Coordination of Governed Agents Over Shared Context: Declarative Governance and Lyapunov-Based Finality

*Submitted to arXiv*

<https://github.com/DealExMachina/swarm-of-governed-agents>

February 27, 2026

## Abstract

The emergence of LLM-powered autonomous agents has exposed the limits of pipeline-based coordination: rigid DAGs cannot accommodate contradiction, evolving evidence, or formal convergence. We propose *declarative governance over shared immutable context*, a paradigm where autonomous agents reason over a common semantic graph, propose state transitions, and reach consensus through formal convergence mechanisms—without predefined execution sequences. Five architectural innovations enable this: (1) an append-only event log and a *bitemporal* CRDT-inspired semantic graph—tracking both valid time (when a fact holds in the world) and transaction time (when the system recorded it)—providing shared, monotonic, temporally auditable context; (2) a pluggable policy engine (YAML and OPA-WASM backends) with XACML combining algorithms and immutable decision records, separating policy from agent implementation; (3) Lyapunov-based finality tracking with five gates—monotonicity, evidence coverage, oscillation detection (lag-1 autocorrelation), quiescence, and minimum content—plus pressure-directed activation and human-in-the-loop routing; (4) Ed25519-signed finality certificates embedding policy-version hashes for cryptographic non-repudiation; (5) three-tier governance routing (deterministic rules, oversight agent, full LLM) with circuit-breaker fallback ensuring continuous operation without LLM availability. Fine-grained access control (OpenFGA) governs agent execution, document access, and policy modification. Unlike orchestration frameworks (AutoGen, CrewAI, LangGraph) that wire agents into chains or graphs, our system lets agents independently reason over shared state and propose actions evaluated by declarative policy. Validation on a realistic M&A due-diligence scenario (Project Horizon) demonstrates convergence in 12 rounds across 5 contradictory documents, with 83% autonomous contradiction resolution, complete audit trail, and structured human review of residual disagreements. Crucially, finality is not terminal: new evidence, regulatory amendments, or periodic review triggers re-open convergence over the same graph, producing a chain of certified checkpoints that models the perpetual lifecycle of regulated knowledge. The resulting architecture is designed for regulated environments where temporal auditability, policy traceability, ongoing monitoring, and cryptographic proof of every decision point are operational requirements.

# 1 Introduction

The rapid maturation of LLM-powered agents—capable of reasoning, planning, and tool use—has created a coordination problem that traditional workflow engines were not designed to solve. Modern agents do not merely execute predefined tasks; they interpret context, generate hypotheses, and produce claims that may contradict each other. This capability demands a fundamentally different coordination model.

Current approaches fall into two categories, both inadequate:

**Pipeline orchestration** (Airflow, Temporal, Prefect) assumes static task graphs executed in topological order. When agents produce contradictory outputs, the pipeline either silently overwrites earlier results or fails entirely. There is no mechanism for tracking disagreement, no formal convergence guarantee, and no governance trail explaining *why* a particular conclusion was reached.

**Agent frameworks** (AutoGen, CrewAI, LangGraph) represent progress: they enable multi-agent conversations, role-based delegation, and tool use. However, they still wire agents into predefined topologies—sequential chains, hierarchical delegation, or cyclic graphs with hardcoded routing. Coordination logic is embedded in code, not governed by policy. There is no shared immutable context, no formal finality mechanism, and no fine-grained access control.

We propose a third path: *declarative governance over shared context*. Rather than sequencing agents, the system maintains:

1. **Shared bitemporal context:** An append-only event log and a CRDT-inspired semantic graph with dual temporality (valid time and transaction time), where confidence scores ratchet upward, contradictions are tracked explicitly, and no state is ever deleted—only superseded. Bitemporal queries enable point-in-time reconstruction for regulatory audit.
2. **Declarative governance with pluggable policy engines:** Agent activation and state transitions are governed by human-readable YAML rules or OPA-WASM compiled Rego policies. No agent directly modifies shared state; all propose transitions that governance evaluates before an executor implements them. XACML combining algorithms resolve multi-policy conflicts. Every decision produces an immutable, policy-version-stamped audit record.
3. **Formal convergence with five gates and perpetual lifecycle:** A Lyapunov disagreement function  $V(t)$  tracks distance to targets across four weighted dimensions. Monotonicity gates, evidence coverage, oscillation detection, quiescence, and minimum-content gates provide layered guarantees against premature finality. When autonomous convergence stalls, the system routes to structured human review with full context. Finality is certified with Ed25519-signed JWS certificates—but finality is not terminal: new evidence, regulatory changes, or periodic review triggers re-open convergence over the same graph, producing a chain of certified checkpoints that models the perpetual lifecycle of regulated knowledge.
4. **Three-tier governance routing:** Deterministic rule evaluation (zero LLM tokens), lightweight oversight agent, and full LLM-backed governance agent, with circuit-breaker fallback ensuring

continuous operation without LLM availability.

5. **Fine-grained access governance:** OpenFGA enforces who can read documents, propose transitions, approve decisions, and modify policies.

This paper presents the design, formal properties, and experimental validation of this system, with particular attention to its fitness for regulated environments.

### 1.1 Contribution and Scope

The paper makes the following concrete contributions:

- An architectural paradigm—declarative governance over shared immutable context—that replaces predefined agent topologies with policy-driven coordination over a common semantic graph.
- A formal convergence mechanism based on a Lyapunov disagreement function with five independent gates (monotonicity, evidence coverage, oscillation detection, quiescence, minimum content) that provides layered guarantees against premature finality.
- A perpetual finality lifecycle in which certified checkpoints—not terminal decisions—model the ongoing nature of regulated knowledge, with Ed25519-signed certificates binding each decision to the exact policy version that produced it.
- A three-tier governance routing architecture (deterministic rules, oversight agent, full LLM) with circuit-breaker fallback, ensuring continuous governance without LLM availability.
- Validation on a realistic M&A due-diligence scenario demonstrating convergence, contradiction resolution, and structured human review.

For clarity, the following are explicit limits of the paper’s claims:

#### What the paper does not demonstrate.

- *Statistical generality.* The Project Horizon validation is a single scenario with synthetic documents; no confidence intervals, hypothesis tests, or distributional claims are supported.
- *Byzantine fault tolerance.* All agents are assumed cooperative. Adversarial agents that inject false claims, manipulate confidence scores, or game the governance protocol are not addressed. Integration with Byzantine-tolerant consensus machinery [7, 2] remains future work.
- *Scalability beyond proof of concept.* Validation covers 50 claims, 5 documents, and 7 agents. Coordination dynamics at production scale (thousands of claims, hundreds of agents) are untested.

- *Machine-checked convergence proofs.* Convergence guarantees are validated empirically through benchmarks and unit tests, not by formal verification tools or proof assistants.
- *Real LLM stochasticity.* Convergence benchmarks use synthetic trajectories. The oscillation detection and trajectory quality mechanisms are designed for real-world stochasticity but have not been validated against it.
- *Protocol self-modification.* Governance rules are declarative but static within a deployment. The system does not implement governed self-modification of its own coordination protocol, as studied by de la Chica Rodriguez and Vera Díaz [3].
- *Multi-scope finality.* Cross-scope governance, hierarchical finality (parent scope depending on child scopes), and inter-scope certificate chains are designed but not validated.

These boundaries are revisited in detail in Section 12.

## 2 Related Work

### 2.1 Pipeline Orchestration

Workflow engines (Apache Airflow, Temporal, Prefect) implement the DAG model: define a static graph, schedule tasks in topological order, apply retry logic on failure. This model is effective for deterministic ETL pipelines but breaks when agents produce conflicting outputs, when new evidence appears mid-execution, or when tasks are interdependent in non-linear ways. None provide formal finality guarantees or governance-as-coordination.

### 2.2 Agent Coordination Frameworks

Recent frameworks enable multi-agent collaboration:

**AutoGen** [18] introduces conversable agents with role-based chat patterns. Coordination is conversation-driven: agents exchange messages in predefined patterns (sequential, group chat, nested). However, coordination topology is hardcoded in Python, there is no shared persistent state between conversations, and no formal convergence mechanism.

**CrewAI** [11] organizes agents into crews with roles, goals, and tools. A hierarchical manager delegates tasks. Coordination follows a fixed process (sequential or hierarchical), with no mechanism for handling contradictions between agent outputs or tracking disagreement over time.

**LangGraph** [8] models agent coordination as state machines with explicit edges and conditional routing. This is closer to our approach—state is explicit and transitions are defined—but routing logic is embedded in code, not declarative policy. There is no immutable audit log, no CRDT semantics, and no formal convergence tracking.

All three frameworks assume that coordination topology can be predefined. Our system replaces topology with policy: agents reason independently over shared state, propose transitions, and governance rules determine what happens next.

## 2.3 Byzantine Consensus and Multi-Agent Coordination

The Byzantine consensus problem, formalized by Lamport, Shostak, and Pease [7], established that agreement among distributed processes is possible if and only if  $n \geq 3f + 1$ , where  $f$  is the number of Byzantine-faulty nodes. Practical Byzantine Fault Tolerance (PBFT) [2] made this feasible in partially synchronous systems with  $O(n^2)$  message complexity. Ren et al. [16] surveyed consensus problems in multi-agent coordination, establishing the graph-theoretic foundations for state agreement under static and time-varying topologies. Zheng et al. [19] introduced confidence-weighted Byzantine fault tolerance (CP-WBFT), incorporating agent reliability into consensus—a direction relevant to LLM-based agents whose outputs carry inherent uncertainty. Mao et al. [10] extended the Byzantine Generals framework to practical multi-agent systems with the Imperfect Byzantine Generals Problem (IBGP), addressing local coordination without full global consensus.

Our system does not implement Byzantine consensus directly; agents are assumed cooperative. However, the semantic graph’s monotonic constraints (confidence ratchets upward, contradictions are irreversible) provide partial robustness against downward manipulation, and the three-tier governance architecture limits the impact of any single agent’s faulty output by requiring governance approval for all state transitions.

## 2.4 Self-Evolving Coordination Protocols

De la Chica Rodriguez and Vera Díaz [3] introduced Self-Evolving Coordination Protocols (SECP): coordination mechanisms that permit bounded, auditable self-modification of their own decision rules while preserving formal invariants. In a controlled feasibility study with six decision modules evaluating six Byzantine consensus proposals, they demonstrated that (i) non-scalar coordination rules can be synthesized by current AI models, (ii) a single governed modification increased proposal coverage by 50% while preserving declared invariants, and (iii) a systematic trade-off exists between protocol coverage (how many proposals are accepted) and evaluator autonomy (the capacity of individual modules to impose non-compensable objections).

Their work is complementary to ours in a precise sense. SECP addresses *protocol-level coordination*: how heterogeneous evaluator judgments are aggregated into accept/reject decisions, and how that aggregation rule can evolve. Our system addresses *state-level coordination*: how agents reason over shared context, how disagreement is tracked formally, and how convergence is certified. SECP’s explicit open questions—multi-iteration convergence dynamics, formal invariant preservation under repeated modification, and audit trail infrastructure—are directly addressed by our Lyapunov-based convergence tracking, five-gate finality mechanism, and bitemporal audit graph. Conversely, their non-scalar coordination with non-compensable objection rights and Byzantine fault tolerance assumptions address gaps we acknowledge: our convergence metric is scalar, our governance rules are static within a deployment, and we do not handle adversarial agents.

A unified architecture combining SECP’s governed protocol evolution with our formal convergence tracking would enable multi-iteration protocol modification where each iteration’s effect on the Lyapunov disagreement function  $V(t)$  is measured, gates prevent premature adoption of harmful

modifications, and the certificate chain records the protocol version under which each decision was made.

## 2.5 Consensus and Convergence Theory

Olfati-Saber and Murray [13] established the Lyapunov consensus framework for multi-agent systems, proving that monotonically decreasing disagreement functions guarantee convergence under mild assumptions. We adapt this to knowledge-work coordination, defining a weighted disagreement function over four semantic dimensions. The classical theory [9] provides the mathematical foundation: if a scalar function  $V \geq 0$  is strictly decreasing along system trajectories and  $V = 0$  only at the equilibrium, the system converges.

Duan et al. [5] introduced the Aegean protocol with monotonicity gates and coordination invariants, requiring  $\beta$  consecutive non-decreasing rounds before declaring convergence. We adopt this mechanism directly.

Camacho et al. [1] proposed EMA-based stagnation detection in the MACI framework, identifying when systems plateau despite appearing active. Our plateau detection mechanism derives from this work.

## 2.6 CRDT Semantics and Monotonic State

Laddad et al. [6] demonstrated CRDT monotonic merge operations (CodeCRDT) that prevent state regression in distributed systems. Our semantic graph adopts this principle: confidence scores ratchet upward only, contradictions once marked are irreversible, and nodes are staled rather than deleted.

## 2.7 Stigmergic Coordination

Dorigo et al. [4] formalized stigmergic coordination in swarm intelligence: agents respond to environmental signals (pheromones) rather than direct communication. Our pressure-directed activation mechanism applies this principle—agents are routed toward high-pressure dimensions (bottleneck areas) without centralized scheduling.

## 2.8 Bitemporal Data Models

Snodgrass [17] established the bitemporal data model: every fact carries two independent time axes—valid time (when true in the world) and transaction time (when recorded by the system). This enables point-in-time reconstruction on either or both axes, a capability required by financial regulations (SOX, IFRS 9) but absent from existing agent frameworks. We apply bitemporal modeling to the semantic graph, enabling temporal contradiction detection and regulatory audit.

## 2.9 Policy Engines and Access Control

The XACML standard [12] defines combining algorithms for multi-policy evaluation (deny-overrides, first-applicable). Open Policy Agent (OPA) [14] provides a general-purpose policy engine with Rego as its policy language and WebAssembly compilation for in-process evaluation. Pang et al. [15] formalized Zanzibar-style relationship-based access control, implemented in OpenFGA. We integrate all three: OPA-WASM as a pluggable policy backend, XACML combining algorithms for multi-policy resolution, and OpenFGA for fine-grained governance of document access, transition approval, and policy modification.

## 3 Problem Setting

This section formalizes the core abstractions of the system. Definitions follow the style of de la Chica Rodriguez and Vera Díaz [3] to facilitate cross-referencing between protocol-level and state-level coordination.

### 3.1 Shared Context Graph

**Definition 1** (Semantic Graph). *Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$  be a directed labeled graph where  $\mathcal{N}$  is a set of typed nodes and  $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N} \times \mathcal{L}$  is a set of labeled edges. Node types are drawn from  $\mathcal{T} = \{\text{CLAIM}, \text{GOAL}, \text{RISK}, \text{ENTITY}\}$ . Edge labels are drawn from  $\mathcal{L} = \{\text{SUPPORTS}, \text{CONTRADICTS}, \text{RESOLVES}, \text{SUPERSEDES}\}$ .*

*Each node  $n \in \mathcal{N}$  carries:*

- a confidence score  $c(n) \in [0, 1]$ ,
- a valid-time interval  $[vf(n), vt(n)]$  (when the fact holds in the world),
- a transaction-time interval  $[ra(n), sa(n)]$  (when the system recorded and possibly superseded it),
- a source provenance  $\sigma(n)$  referencing the originating document and agent.

**Definition 2** (Monotonicity Constraints). *The graph  $\mathcal{G}$  satisfies three monotonicity invariants:*

1. **Confidence ratchet:** For any node  $n$ , if  $c(n) = x$  at transaction time  $t_1$ , then for all  $t_2 > t_1$  where  $n$  is not superseded,  $c(n) \geq x$ .
2. **Irreversible contradictions:** If  $(n_1, n_2, \text{CONTRADICTS}) \in \mathcal{E}$  at transaction time  $t$ , the edge persists for all  $t' > t$ . Resolution requires adding a new RESOLVES edge, not deleting the contradiction.
3. **Staling, not deletion:** No node or edge is removed from  $\mathcal{G}$ . Superseded nodes receive a finite  $sa(n)$  timestamp; the current view filters to  $sa(n) = \infty$ .

*These invariants ensure that  $\mathcal{G}$ , restricted to the current view, evolves monotonically toward resolution.*

### 3.2 Governance Function

**Definition 3** (Proposal and Governance). Let  $\mathcal{A} = \{a_1, \dots, a_n\}$  be a set of agents. Each agent  $a_i$  observes the current view of  $\mathcal{G}$  and emits a proposal  $p = (\delta, j, a_i)$  where  $\delta$  is a candidate state transition (a set of node/edge additions or confidence updates satisfying Definition 2),  $j$  is a natural-language justification, and  $a_i$  is the proposing agent.

A governance function  $\Pi$  is a deterministic mapping:

$$\Pi : \mathcal{P} \times \mathcal{G} \times \mathcal{R} \longrightarrow \{\text{APPROVE}, \text{REJECT}, \text{ESCALATE}\}$$

where  $\mathcal{P}$  is the proposal space,  $\mathcal{G}$  is the current graph state, and  $\mathcal{R}$  is the active rule set (YAML or compiled Rego policy). Only proposals mapped to APPROVE are applied to  $\mathcal{G}$  by the executor. ESCALATE routes to human review.

The central system invariant is: no agent directly modifies  $\mathcal{G}$ . All modifications pass through  $\Pi$ .

### 3.3 Convergence and Finality

**Definition 4** (Lyapunov Disagreement Function). Let  $\mathcal{D} = \{d_1, \dots, d_k\}$  be a set of convergence dimensions, each with a weight  $w_d > 0$  ( $\sum_d w_d = 1$ ), a target value  $\tau_d$ , and a measurement function  $\mu_d : \mathcal{G} \rightarrow [0, 1]$ . The Lyapunov disagreement function is:

$$V(t) = \sum_{d \in \mathcal{D}} w_d \cdot (\tau_d - \mu_d(\mathcal{G}_t))^2$$

where  $\mathcal{G}_t$  is the graph state at discrete time  $t$  (the  $t$ -th governance cycle). By construction,  $V(t) \geq 0$ , and  $V(t) = 0$  if and only if all dimensions have reached their targets.

**Definition 5** (Finality Predicate). Let  $S(t) = 1 - V(t)/V_{\max}$  be the normalized goal score. Define five gate predicates:

$$\begin{aligned} G_A(t) &= \bigwedge_{i=0}^{\beta-1} [S(t-i) \geq S(t-i-1) - \epsilon_m] && (\text{monotonicity, } \beta = 3, \epsilon_m = 0.001) \\ G_B(t) &= \text{EvidenceCoverage}(\mathcal{G}_t) \wedge (\text{ContradictionMass}(\mathcal{G}_t) = 0) && (\text{evidence + contradiction}) \\ G_C(t) &= (Q(t) \geq 0.7) && (\text{trajectory quality, no oscillation}) \\ G_D(t) &= (\text{IdleCycles}(t) \geq \gamma) \wedge (\text{IdleTime}(t) \geq \omega) && (\text{quiescence}) \\ G_E(t) &= (|\mathcal{N}_t| > 0) \wedge (|\text{Goals}(\mathcal{G}_t)| > 0) && (\text{minimum content}) \end{aligned}$$

The finality predicate is:

$$F(t) = [S(t) \geq \theta_{auto}] \wedge \bigwedge_{i \in \{A,B,C,D,E\}} G_i(t)$$

where  $\theta_{auto} = 0.92$ . When  $F(t) = 1$ , the system transitions to RESOLVED and issues a signed finality

*certificate. When  $F(t) = 0$  but  $S(t) \geq \theta_{hitl}$ , a human-in-the-loop review is triggered.*

### 3.4 Threat Model and Assumptions

The current system operates under the following assumptions:

1. **Cooperative agents.** All agents are assumed to operate in good faith within their governance constraints. No Byzantine, adversarial, or colluding agents are modeled.
2. **Trusted governance layer.** The policy engine, epoch-based CAS, and finality evaluator are assumed correct. Formal machine-checked verification of these components is not provided.
3. **Trusted infrastructure.** Postgres, NATS, and OpenFGA are assumed to function correctly. Network partitions and service degradation are handled by circuit breakers and graceful degradation, not by Byzantine fault tolerance.
4. **Single-scope operation.** Cross-scope governance and hierarchical finality are designed but not validated.

These assumptions are substantially weaker than those required for Byzantine consensus ( $n \geq 3f+1$ ) [7]. Relaxing them—particularly the cooperative-agent assumption—is the primary direction for future work (Section 13).

### 3.5 Complexity Bounds

The three-node state cycle constrains the coordination topology. In each cycle (`ContextIngested` → `FactsExtracted` → `DriftChecked`), each of  $n$  agents is activated at most once (activation filters enforce this). The message complexity per cycle is therefore  $O(n)$ : one proposal per agent, one governance evaluation per proposal, one execution per approved proposal. Over  $k$  cycles to convergence, total message complexity is  $O(nk)$ .

This contrasts with classical BFT protocols requiring  $O(n^2)$  messages per consensus round [2]. The reduced complexity follows from the cooperative-agent assumption: without Byzantine faults, the system does not need all-to-all message exchange for agreement. Instead, agreement is mediated by the shared graph  $\mathcal{G}$  and the governance function  $\Pi$ .

The number of cycles  $k$  to convergence is bounded above by the EXPIRED timeout (configurable, default 30 days) and below by the minimum  $\beta + \gamma$  rounds required by Gates A and D. In the Project Horizon validation,  $k = 12$ .

## 4 Core Design

### 4.1 Design Principle: Proposals, Approval, Execution

The central invariant of the system is: *no agent directly modifies shared state*. Instead:

1. **Agents propose:** Each agent reads shared context, reasons about it, and emits a proposal (a candidate state transition with justification).
2. **Governance evaluates:** Declarative rules (YAML) determine whether the proposal is approved, blocked, or routed to human review.
3. **Executor implements:** Only approved proposals are applied to shared state, via atomic epoch-based compare-and-swap (CAS) transitions.

This pattern provides complete auditability: every state change has a proposal, an evaluation, and an execution record. It also enables governance without code changes: rules are configuration, not compiled logic.

## 4.2 Shared Immutable Context

All agents read from and write to two shared data structures:

**Append-Only Event Log (Context WAL).** Every claim, fact, and agent decision is recorded as an immutable event in a Postgres write-ahead log:

```
{
  "timestamp": "2025-08-15T10:32:00Z",
  "agent_id": "facts_extraction_v2",
  "event_type": "claim_emitted",
  "payload": {
    "entity": "NovaTech AG",
    "relation": "annual_recurring_revenue",
    "value": "EUR 50M",
    "confidence": 0.92,
    "source_doc": "analyst_briefing.pdf"
  }
}
```

Events are never modified or deleted. This provides tamper-proof auditability, full reproducibility (replay the log to reconstruct any state), and causal traceability (which facts triggered which agent actions).

**CRDT-Inspired Semantic Graph.** A Postgres + pgvector database maintains semantic relationships with monotonic guarantees [6]:

- **Monotonic confidence:** Confidence scores ratchet upward only. A claim at 0.85 can become 0.92 but never revert to 0.80.
- **Irreversible contradictions:** Once a contradiction edge is created between two claims, it cannot be deleted—only resolved by creating a resolution edge.

- **Staling, not deletion:** Superseded nodes are marked stale, preserving the full reasoning history.

These CRDT semantics prevent state regression and enable formal convergence guarantees: the semantic graph can only move toward resolution, never away from it.

**Bitemporal Semantic Graph.** Beyond monotonic merge, the semantic graph implements *dual temporality* [17]: every node and edge carries two independent time axes.

- **Valid time** (`valid_from`, `valid_to`): the interval during which a fact holds in the real world. A revenue figure may be valid for fiscal year 2024; a compliance certificate valid until its expiry date.
- **Transaction time** (`recorded_at`, `superseded_at`): when the system learned or corrected the fact. A node superseded at  $t_s$  remains in the graph with `superseded_at` =  $t_s$ ; its replacement carries `recorded_at` =  $t_s$ .

The *current view* filter—`superseded_at IS NULL ∧ (valid_to IS NULL ∨ valid_to > now())`—restricts queries to facts that are both temporally valid and not yet corrected. As-of queries on either axis or both enable point-in-time reconstruction:

- **As-of valid time:** “What was believed true on reporting date  $T$ ? ”
- **As-of transaction time:** “What did the system know at audit date  $T'$ ? ”
- **Combined:** “What did the system know at  $T'$  about facts valid at  $T$ ? ”

Contradiction detection respects valid-time overlap: two claims contradict only if their validity windows intersect. A revenue figure valid in Q1 does not contradict a revised figure valid from Q2 onward. This prevents false contradictions from temporal misalignment and enables the finality evaluator to correctly assess unresolved disagreements.

An *evidence schema* declares required evidence types and maximum staleness per domain. Gate B of the finality evaluator (Section 5.5) uses this schema to block finality when required evidence is missing or has exceeded its temporal validity—ensuring that decisions are not finalized on stale data.

### 4.3 Declarative Governance

Agent activation and state transitions are defined in human-readable YAML. The actual governance configuration separates three concerns: an approval mode, drift-triggered policy rules, and transition-blocking rules:

```

mode: YOLO      # YOLO | MITL | MASTER (per-scope overrides below)

rules:
  - when:
    drift_level: [medium, high]
    drift_type: contradiction
    action: open_investigation
  - when:
    drift_level: [high]
    drift_type: entropy
    action: halt_and_review

transition_rules:
  - from: DriftChecked
    to: ContextIngested
    block_when:
      drift_level: [critical]
    reason: "Critical drift blocks cycle reset"

```

Three governance modes provide different compliance profiles:

- **YOLO**: Automatic approval for valid transitions. Suitable for low-risk, internal workflows.
- **MITL (Man-in-the-Loop)**: All proposals route to human review queue. Required for standard compliance workflows.
- **MASTER**: Deterministic rule-based decisions without LLM rationale. Suitable for highest-sensitivity scenarios where LLM non-determinism is unacceptable.

Per-scope overrides allow mixing modes within a single system (e.g., YOLO for low-risk extraction, MITL for financial claims).

**Policy Engine Architecture.** Behind the YAML surface, a pluggable `PolicyEngine` interface decouples governance semantics from evaluation backend. Two implementations are provided:

- **YAML engine** (default): evaluates the governance YAML rules directly. No external dependencies; suitable for development and single-team deployments.
- **OPA-WASM engine** [14]: loads compiled Rego policies as WebAssembly modules and evaluates them in-process. This enables enterprise policy teams to author governance rules in Rego, compile them with `opa build`, and deploy without modifying agent code.

When multiple policy backends contribute to a single decision, XACML-inspired combining algorithms [12] resolve conflicts: `deny-overrides` (any deny wins) and `first-applicable` (ordered evaluation) are implemented. This mirrors the Policy Decision Point (PDP) / Policy Enforcement Point (PEP) architecture standard in enterprise access management.

**Decision Records.** Every governance evaluation produces an immutable `DecisionRecord` persisted to a dedicated audit table:

```
{  
  "decision_id": "a3f8c...",  
  "timestamp": "2025-08-15T10:32:01Z",  
  "policy_version": "sha256:7e4f2a...",  
  "result": "allow",  
  "reason": "no blocking rule",  
  "obligations": [],  
  "binding": "yaml"  
}
```

The `policy_version` is a content hash of the governance and finality configuration files at evaluation time, binding each decision to the exact rule set that produced it. This enables post-hoc queries such as: “Was this decision made under the pre-amendment or post-amendment compliance rules?”

**Obligation Enforcement.** Policy rules may attach *obligations*—mandatory post-decision actions (e.g., `dual_review`, `compliance_notification`). An obligation enforcer with a registry-based handler architecture executes these after each decision, ensuring that procedural requirements (notifications, secondary reviews, audit entries) are not merely recommended but enforced as part of the governance pipeline.

## 4.4 Three-Tier Governance Routing

A critical design constraint for regulated environments is that governance must function even when LLM services are degraded or unavailable. The system implements three tiers of governance evaluation, each a strict superset of the previous:

1. **Deterministic evaluation (Tier 1).** Every proposal is first evaluated with zero LLM tokens: the policy engine checks transition rules and drift conditions; OpenFGA checks agent permissions. This produces a deterministic outcome (approve, reject, or pending) with a structured reason. In MASTER mode or when no LLM is configured, this is the final decision.
2. **Oversight agent (Tier 2).** In YOLO mode with an LLM available, a lightweight oversight agent receives the deterministic result and chooses one of three actions: *accept* the deterministic result as-is, *escalate to full LLM* for richer reasoning, or *escalate to human* (MITL). This routing decision is itself audited via the `GovernancePath` field in the context WAL.
3. **Full governance agent (Tier 3).** When the oversight agent escalates, a full LLM-backed governance agent reasons over state, drift, governance rules, and policy checks using tool calls, then publishes an approval or rejection with natural-language rationale.

A *circuit breaker* (3 consecutive failures, 60-second cooldown) protects against LLM service degradation: when the breaker opens, Tiers 2 and 3 are bypassed and the system falls back to Tier 1 deterministic evaluation. This guarantees that governance never stalls due to LLM unavailability—a requirement in environments where agent downtime has compliance implications.

Every decision records which tier produced it (`processProposal`, `oversight_acceptDeterministic`, `oversight_escalateToLLM`, `oversight_escalateToHuman`, or `processProposalWithAgent`), providing auditors with the exact governance path for each state transition.

## 4.5 Three-Node State Cycle

Rather than an open-ended state machine, the system operates on a tight three-node cycle with epoch-based CAS:

$$\text{ContextIngested} \rightarrow \text{FactsExtracted} \rightarrow \text{DriftChecked} \rightarrow \text{ContextIngested}$$

Each transition requires governance approval. Only one agent succeeds per cycle (atomic epoch check). This prevents race conditions and ensures every cycle produces a complete audit record.

## 4.6 Fine-Grained Access Governance (OpenFGA)

OpenFGA [15] enforces relationship-based access control at four levels:

- **Document access:** Only authorized agents and users read sensitive materials.
- **Transition approval:** Governance rules determine who can approve which transitions.
- **Policy modification:** Changes to governance rules require role-based sign-off and version control.
- **Audit access:** Sensitivity-appropriate visibility into audit logs.

## 5 Convergence Theory and Finality

### 5.1 The Finality Problem

Traditional systems declare finality by threshold: “if confidence > 0.95, done.” This is brittle: contradictions may emerge after the threshold is crossed, the system may cycle indefinitely, and there is no formal guarantee of stabilization. We replace threshold-based finality with *stateful convergence tracking*.

### 5.2 Lyapunov Disagreement Function

Following Olfati-Saber and Murray [13], we define a scalar disagreement function  $V(t) \geq 0$  measuring distance to convergence targets:

$$V(t) = \sum_{d \in \mathcal{D}} w_d \cdot (\text{target}_d - \text{actual}_d(t))^2$$

where  $\mathcal{D}$  comprises four weighted dimensions:

Dimension	Target	Weight	Meaning
Claim confidence	$\geq 0.85$	0.30	Active claims at sufficient confidence
Contradiction resolution	$= 0$	0.30	Zero unresolved contradictions
Goal completion	$\geq 0.90$	0.25	Completion ratio of declared goals
Risk score inverse	$< 0.20$	0.15	Residual risk below threshold

Table 1: Convergence dimensions with weights and targets.

### 5.3 Convergence Guarantee Under Monotonic Constraints

The CRDT-inspired monotonicity invariants (Definition 2) provide the foundation for a convergence argument. We state this as a proposition with a proof sketch; a fully machine-checked proof is deferred to future work.

**Proposition 1** (Monotonic Progress Under CRDT Constraints). *Let  $\mathcal{G}_t$  be the semantic graph at cycle  $t$ , and let  $V(t)$  be the Lyapunov disagreement function (Definition 4). If every governance-approved transition  $\delta_t$  applied at cycle  $t$  satisfies the monotonicity constraints of Definition 2, and if at least one of the following holds:*

1. a claim confidence  $c(n)$  increases for some  $n \in \mathcal{N}_t$ ,
2. a contradiction edge receives a RESOLVES edge,
3. a goal is marked complete,
4. a risk score decreases,

then  $V(t+1) \leq V(t)$ , with strict inequality when the affected dimension has nonzero weight and the target has not yet been reached.

*Proof sketch.* Each convergence dimension  $d$  contributes  $w_d \cdot (\tau_d - \mu_d(\mathcal{G}_t))^2$  to  $V(t)$ . The monotonicity constraints ensure that no approved transition can decrease  $\mu_d$  for any dimension:

- *Claim confidence* (dimension 1): the confidence ratchet guarantees  $c(n)$  is non-decreasing, so the fraction of claims above threshold is non-decreasing.
- *Contradiction resolution* (dimension 2): contradictions are irreversible and resolutions are additive, so the unresolved count is non-increasing.
- *Goal completion* (dimension 3): goals are completed monotonically (no un-completion).

- *Risk score* (dimension 4): risk is computed from unresolved contradictions and missing evidence, both of which are non-increasing under approved transitions.

Since  $\mu_d(\mathcal{G}_{t+1}) \geq \mu_d(\mathcal{G}_t)$  for all  $d$ , and  $\tau_d \geq \mu_d(\mathcal{G}_t)$  when targets are not yet met, each squared term is non-increasing. When at least one dimension makes strict progress (conditions 1–4), the corresponding term strictly decreases, giving  $V(t+1) < V(t)$ .  $\square$

**Corollary 1** (Bounded Convergence Time). *Under the conditions of Proposition 1, if every governance cycle produces at least one approved transition making strict progress on some dimension, then  $V(t) \rightarrow 0$  in at most  $K$  cycles, where  $K$  is bounded by the product of the number of claims, contradictions, goals, and risk factors in the initial graph. In practice,  $K$  is further bounded by the EXPIRED timeout.*

Note that this guarantee does *not* hold when agents fail to make progress (the stalled regime,  $\alpha \approx 0$ ). In that case, plateau detection (Gate C) and human-in-the-loop routing provide operational safeguards rather than formal guarantees. The proposition also does not address adversarial agents that could attempt to inflate confidence scores without genuine evidence—a limitation addressed by the threat model (Section 3).

## 5.4 Convergence Rate and ETA

The convergence rate  $\alpha$  is computed as:

$$\alpha = -\ln\left(\frac{V(t)}{V(t-1)}\right)$$

with estimated time to arrival:

$$\text{ETA} = \left\lceil \frac{-\ln(\epsilon/V(t))}{\alpha} \right\rceil, \quad \epsilon = 0.005$$

Three regimes:

- $\alpha > 0$ : Converging. Finite ETA.
- $\alpha \approx 0$ : Stalled. Plateau detection activates.
- $\alpha < -0.05$ : Diverging. Triggers ESCALATED state.

## 5.5 Finality Gates

Auto-finality (RESOLVED) requires passing all five gates simultaneously. Each gate addresses a distinct failure mode of premature finality.

**Gate A: Monotonicity [5].** Goal score must remain non-decreasing for  $\beta = 3$  consecutive rounds before auto-finality eligibility. A single drop  $> 0.001$  resets the counter. This prevents declaring convergence during transient spikes.

**Gate B: Evidence Coverage and Contradiction Mass.** An evidence schema (Section 4.2) declares required evidence types per domain and maximum staleness (`max_age_days`). Gate B blocks finality when required evidence types are missing or when evidence has exceeded its temporal validity. Contradiction mass—the weighted count of unresolved contradictions—must also be zero for auto-resolution. This ensures that finality reflects substantive coverage, not merely the absence of detected problems.

**Gate C: Oscillation Detection and Trajectory Quality.** Gate C addresses a subtle failure mode: a system whose goal score oscillates around the finality threshold without genuinely converging. Two statistical mechanisms detect this:

1. **Direction-change counting:** In a sliding window of the last 10 goal-score evaluations, the number of direction reversals (positive-to-negative or vice versa, with a 0.001 dead band) is counted. Two or more reversals flag oscillation.
2. **Lag-1 autocorrelation:** The Pearson correlation between the goal-score series and its one-step lag is computed. Negative autocorrelation ( $r_1 < -0.3$ ) confirms alternating behavior characteristic of oscillation.

A *trajectory quality score*  $Q \in [0, 1]$  synthesizes both signals:

$$Q = \max(0, 1 - 0.12 \cdot \min(\text{direction\_changes}, 5))$$

with further reduction to  $Q \leq 0.65$  when  $r_1 < -0.3$  and to  $Q \leq 0.85$  on spike-and-drop (latest score well below window maximum). Auto-RESOLVED requires  $Q \geq 0.7$ .

**Gate D: Quiescence.** Configurable via `idle_cycles_min` and `window_ms` in finality configuration. When enabled, RESOLVED is blocked unless the scope has been idle (no state transitions) for the configured number of cycles and time window. This prevents premature finality during active processing bursts where the score happens to cross the threshold momentarily.

**Gate E: Minimum Content.** When all dimensions score 1.0 vacuously—because there are zero claims, zero goals, and zero risks in the graph—Gate E blocks auto-finality. An empty or trivially-seeded scope cannot be declared resolved. This prevents the degenerate case where a newly initialized scope immediately satisfies all threshold conditions.

**Pressure-Directed Activation [4].** Agents are routed toward the dimension with highest residual gap (bottleneck), without centralized scheduling. This stigmergic mechanism ensures resources concentrate where they matter most.

**Divergence Escalation.** If  $\alpha < -0.05$  (disagreement increasing), the system transitions to ESCALATED state, requiring human intervention.

**Human-in-the-Loop Review.** When goal score is in  $[0.40, 0.92)$  and plateau is detected, the system queues a structured HITL review with: dimension breakdown, blocking factors, LLM-generated explanation, and suggested actions (approve finality, provide resolution, escalate, or defer).

## 5.6 Finality States

State	Trigger	Action
RESOLVED	Score $\geq 0.92 +$ Gates A–E all passed	JWS certificate issued
ACTIVE	Score $< 0.92$ or any gate unsatisfied	Continue processing
ESCALATED	Risk $\geq 0.75$ or $\geq 3$ contradictions or $\alpha < -0.05$	Human intervention
BLOCKED	$\geq 5$ idle cycles + $\geq 300$ s without updates + $\geq 1$ contradiction	Stalled
EXPIRED	No activity for 30 days	Process expired
HITL Review	Score $\in [0.40, 0.92)$ + plateau or gate failure	Structured human decision

Table 2: Finality states and their triggers. RESOLVED requires all five gates (monotonicity, evidence coverage, trajectory quality, quiescence, minimum content) plus the auto-finality threshold. Upon RESOLVED, an Ed25519-signed finality certificate is emitted (Section 5.7).

## 5.7 Finality Certificates

When finality is reached—either through automatic convergence (all gates passed) or human approval—the system issues a cryptographically signed *finality certificate* providing non-repudiable proof of the decision.

The certificate payload contains:

```
{
  "scope_id": "project-horizon",
  "decision": "RESOLVED",
  "timestamp": "2025-08-15T14:22:00Z",
  "policy_version_hashes": {
    "governance": "sha256:7e4f2a...",
    "finality": "sha256:b3c91d..."
  },
  "dimensions_snapshot": {
    "claim_confidence": 0.94,
    "contradiction_resolution": 1.0,
    "goal_completion": 0.92,
    "risk_score_inverse": 0.88
  }
}
```

This payload is signed as a compact JWS (JSON Web Signature) using Ed25519 (EdDSA), producing a base64url-encoded triple: `header.payload.signature`. Verification requires only the

public key and no system access, enabling external auditors, counterparties, or regulators to independently validate that:

1. A specific scope reached a specific finality decision;
2. The decision was made at a specific timestamp;
3. The governance and finality rules in effect were identified by content hashes, binding the decision to an exact, reproducible rule set;
4. The dimensional scores at the moment of finality are recorded.

Certificates are persisted in a dedicated `finality_certificates` table and exposed via API for retrieval. Key management supports both environment-provided PEM keys (production) and ephemeral key pairs (development).

For regulated environments, finality certificates serve as machine-verifiable sign-off records—the agent-system equivalent of an authorized signatory’s approval, with the added property that the policy version under which the decision was made is cryptographically bound to the certificate.

## 5.8 Perpetual Finality: From Checkpoint to Checkpoint

A critical distinction separates this architecture from systems that treat finality as terminal. In most regulated domains, knowledge does not reach a permanent resting state. New documents arrive, regulations are amended, market conditions shift, periodic reviews are mandated, and previously resolved contradictions may re-emerge under new evidence. Finality here is not the end of a process—it is a *certified checkpoint* within an indefinite lifecycle.

The shared context graph enables this directly. When a scope reaches RESOLVED:

1. A finality certificate is issued and persisted, recording the decision, the dimensional scores, the policy version, and the timestamp.
2. The semantic graph is *not archived or frozen*. It remains the live, authoritative representation of knowledge for that scope.
3. When new context arrives—a new document ingested into the WAL, a regulatory change requiring re-evaluation, or a scheduled periodic review trigger—the scope re-enters ACTIVE state.
4. The new convergence cycle starts from the *existing graph state*: all prior claims, contradictions, resolutions, and confidence scores are preserved. New facts layer on top. New contradictions may form between new and old claims.
5. The Lyapunov function  $V(t)$  is recomputed. If new contradictions have increased disagreement,  $V$  rises and convergence must recur. If the new information merely confirms or refines prior conclusions,  $V$  remains low and finality may be reached quickly.

6. A new finality certificate is issued when convergence is re-achieved, creating a *chain of certificates*: each certifies the scope’s state at a specific point in time, under a specific policy version, with a specific dimensional snapshot.

Bitemporal state is what makes this work without information loss. Prior facts are never deleted. When a new document supersedes a previous claim (e.g., restated financials), the old node receives a `superseded_at` timestamp; the new node carries its own `valid_from`. The full temporal history is preserved: an auditor can reconstruct what the graph looked like at any prior finality point, compare it to the current state, and see exactly which new information triggered the re-evaluation.

This architecture models the operational reality of regulated knowledge work:

- In M&A due diligence, the initial close produces one finality certificate. Post-merger integration monitoring ingests quarterly reports that may introduce new contradictions—producing a new convergence cycle and a new certificate each quarter.
- In ongoing KYC/AML, an initial customer risk assessment reaches finality. Periodic reviews (annual for standard risk, quarterly for high risk) re-inject updated information, triggering re-convergence against the same graph.
- In pharmaceutical regulation, an initial marketing authorization reaches finality. Post-market pharmacovigilance data feeds continuously into the context graph, potentially re-opening finality when adverse events contradict prior safety claims.

The system does not merely *support* this lifecycle; it is the natural operating mode. The three-node cycle (`ContextIngested` → `FactsExtracted` → `DriftChecked`) runs indefinitely, and finality evaluation runs after every governance decision. What changes between cycles is the graph’s content, not its structure.

## 6 Agent Architecture

Seven agents operate independently within governance constraints, gated by deterministic activation filters that consume zero LLM tokens:

**Facts Agent.** Extracts claims, goals, and risks from unstructured context using an LLM pipeline with optional GLiNER2 named-entity recognition and NLI-based contradiction detection. Outputs typed nodes to the semantic graph with confidence scores, source references, and optional valid-time intervals. Activated by a `sequence_delta` filter: runs only when new events appear in the context WAL since the last extraction.

**Drift Agent.** Computes changes from baseline across four drift types (contradiction, goal misalignment, factual inaccuracy, entropy). Detects both temporal and intra-batch contradictions.

Classifies severity (none/low/medium/high) and routes to governance for policy evaluation. Activated by a `hash_delta` filter: runs only when the facts artifact has changed since the last drift analysis.

**Planner Agent.** Synthesizes facts and drift into ranked proposals. Maps drift dimensions to required next actions, considers pressure-directed routing to bottleneck dimensions, and proposes state transitions with justification. Activated by a `hash_delta` filter on the drift artifact.

**Status Agent.** Tracks  $V(t)$ ,  $\alpha$ , monotonicity gate, and plateau detection. Produces executive briefings. Routes near-finality cases to HITL review with full convergence analysis. Activated by a `timer` filter with configurable short and full intervals.

**Governance Agent.** Implements the three-tier governance routing described in Section 4.4. Tier 1 deterministic evaluation checks the policy engine (YAML or OPA-WASM) and OpenFGA permissions. Tier 2 oversight agent routes to accept, LLM escalation, or human escalation. Tier 3 full LLM agent reasons over state and drift with tool access. A circuit breaker (3 failures, 60s cooldown) ensures fallback to Tier 1 on LLM degradation. Every decision records its governance path for audit.

**Executor.** Implements approved actions via atomic state transitions with epoch CAS. Records in append-only Context WAL. Updates semantic graph with monotonic upserts. Handles both deterministic execution and LLM-backed execution (with fallback).

**Tuner Agent.** Periodically optimizes activation filter parameters (cooldown intervals, sensitivity thresholds, minimum event counts) based on accumulated statistics: activation count, productive vs. wasted activations, average latency. Filter configurations are versioned and snapshotted to S3 for audit. This creates a closed optimization loop: filter stats reveal whether agents are being activated productively, and the tuner adjusts parameters to minimize wasted LLM invocations.

**Activation filters.** Each agent’s activation is gated by a deterministic filter stored in Postgres with four types: `sequence_delta` (new WAL events), `hash_delta` (artifact content changed), `timer` (time elapsed), and `pressure_directed` (convergence pressure exceeds threshold for the agent’s dimension). Filters prevent redundant agent invocations, reducing LLM token consumption. A `pressure_directed` filter consults the latest convergence history and activates the agent only if its associated dimension (e.g., `contradiction_resolution` for the drift agent) has the highest or near-highest pressure—implementing stigmergic routing at the activation layer.

**Key invariant:** Agents do not call each other directly. They emit events that trigger governance rules. This decoupling enables resilience (agent failure does not cascade), auditability (every interaction is logged), and scalability (agents can be independently scaled via hatchery-based dynamic provisioning using M/M/c queueing theory [4]).

## 7 Comparison with Agent Frameworks

Property	AutoGen	CrewAI	LangGraph	SECP	Ours
Coordination model	Chat	Hierarchical	State machine	Protocol agg.	Decl. policy
Shared persistent state	No	No	Partial	No	Yes
Bitemporal state	No	No	No	No	Yes
Formal convergence	No	No	No	No <sup>†</sup>	Yes (Lyapunov)
Contradiction tracking	No	No	No	Implicit <sup>‡</sup>	Yes (graph)
Immutable audit log	No	No	No	Partial	Yes
Non-scalar coordination	No	No	No	Yes	Partial*
Protocol self-modification	No	No	No	Yes	No
Byzantine fault model	No	No	No	Yes	No
Declarative governance	No	No	No	No	Yes (YAML+OPA)
Cryptographic finality	No	No	No	No	Yes (Ed25519)
Fine-grained access control	No	No	No	No	Yes (OpenFGA)
CRDT monotonic semantics	No	No	No	No	Yes
HTL finality	Manual	Manual	Manual	No	Structured
Agent topology	Predefined	Predefined	Predefined	Fixed	Emergent
LLM-free fallback	No	No	No	N/A	Yes (Tier 1)

Table 3: Comparison with agent coordination frameworks and SECP [3]. <sup>†</sup>SECP demonstrates single-step modification but not multi-step convergence. <sup>‡</sup>Module disagreement is tracked through non-scalar objections but not in a persistent graph. \*Our Lyapunov function is scalar; per-dimension analysis is available but finality uses an aggregate threshold.

The comparison reveals two orthogonal innovation axes. Existing agent frameworks (AutoGen, CrewAI, LangGraph) *wire agents into topologies* where coordination is embedded in code; neither addresses formal convergence, governance auditability, or protocol adaptability. SECP [3] addresses the protocol-level question—how to aggregate heterogeneous evaluator judgments with non-compensable objection rights and governed self-modification—but lacks shared persistent state, formal convergence tracking, or cryptographic finality. Our system addresses the state-level question—how agents reason over shared context and converge formally—but uses a scalar convergence metric and does not support protocol self-modification.

The complementarity is precise: SECP’s non-scalar coordination with coverage-autonomy navigation could replace or augment our scalar governance function  $\Pi$ , while our Lyapunov convergence tracking and bitemporal audit infrastructure could provide the multi-iteration foundation that SECP’s single-shot design currently lacks. Neither system alone addresses all requirements of governed multi-agent coordination in regulated environments; a synthesis would.

This distinction matters for enterprise adoption: when business requirements change (new compliance rules, additional risk factors, different approval workflows), our system requires YAML edits. Existing frameworks require code changes, testing, and redeployment.

## 8 Validation: Project Horizon

### 8.1 Scenario: M&A Due Diligence

Project Horizon simulates due diligence on NovaTech AG, a B2B SaaS firm in supply chain compliance. Five documents are ingested sequentially, each introducing new facts and contradictions:

1. **Analyst Briefing:** Baseline facts (EUR 50M ARR, 7 patents, 45% CAGR). Low finality score ( $\sim 0.15\text{--}0.30$ ).
2. **Financial DD:** ARR revised to EUR 38M—a EUR 12M discrepancy. High contradiction drift triggers governance block, preventing the cycle from advancing silently.
3. **Technical Assessment:** CTO and 2 senior engineers departing. Medium drift escalates to risk committee review.
4. **Market Intelligence:** Patent infringement suit filed. Multiple unresolved contradictions push toward ESCALATED finality state.
5. **Legal Review:** Partial contradiction resolution. Goal score crosses 0.75 (near-finality) but remains below 0.92 (auto-finality). HITL review queued with structured decision options: approve finality, provide resolution, escalate, or defer.

### 8.2 Key Governance Moments

- **Cycle blocked after Document 2:** Financial discrepancy is not silently absorbed. Governance rules prevent advancement until contradiction is addressed.
- **Investigation recommended:** Contradiction drift signals formal investigation.
- **Risk escalation:** Critical personnel departures routed to structured escalation path.
- **Near-finality HITL:** System has sufficient data to recommend but insufficient confidence for autonomous decision. Human receives dimension breakdown, blockers, and suggested actions.

### 8.3 Results

Metric	Value
Input documents	5 (analyst, financial, technical, market, legal)
Facts extracted	312
Contradictions identified	18
Agent-resolved contradictions	15 (83%)
Contradictions routed to human	3 (17%)
Total convergence rounds	12
Wall-clock time	240 sec
Immutable audit events	487
Policy rule evaluations	156
Governance blocks triggered	3

Table 4: Project Horizon results.

### 8.4 Testing and Benchmarks

**197 unit tests** across 26 files covering: state graph CAS transitions (11), convergence tracker stability (28), finality evaluator logic (11), governance agent modes (18), embedding pipeline (11), and supporting infrastructure.

**7 convergence benchmarks** validating mathematical properties independently of infrastructure:

Scenario	Rounds	Outcome
Steady convergence ( $\sim 5\%/\text{round}$ )	15	Monotonic, converging, $\text{ETA} \approx 0$
Plateau at 0.70 ( $\pm 0.002$ oscillation)	10	Stagnation detected
Spike-and-drop ( $0.95 \rightarrow 0.70$ )	–	Monotonicity gate blocks premature finality
Divergence (contradictions increase)	–	Negative $\alpha$ , zero forward progress
One-dimension bottleneck	–	Pressure identifies blocker dimension
Fast convergence ( $\geq 0.92$ in 3 rounds)	3	No false plateau during rapid improvement
Empty graph (no claims/goals)	–	Safe defaults, no division by zero

Table 5: Convergence benchmarks validating formal properties.

### 8.5 Known Validation Gaps

The project explicitly acknowledges: no stress testing for high throughput, no chaos engineering for service failures, single-scope E2E only, synthetic convergence data (not real LLM trajectories), and unmeasured LLM oversight quality.

## 9 Proposed Experimental Protocol

The following experiments are designed to address the open questions identified in this work and in the SECP feasibility study [3]. They are specified with sufficient detail for independent reproduction.

Together, they target five axes: convergence dynamics, scalability, finality robustness, multi-level governance, and the coverage–autonomy trade-off.

### 9.1 Experiment 1: Multi-Iteration Convergence Dynamics

**Objective.** Characterize the trajectory of  $V(t)$  over extended convergence cycles with incremental evidence injection—the multi-iteration dynamics that SECP [3] explicitly could not study with their single-shot design.

**Protocol.**

1. Initialize a scope with a seed document producing  $\sim 20$  claims.
2. Over 20 convergence cycles, inject one new document per cycle, each introducing  $c \in \{0, 1, 3, 5\}$  contradictions with existing claims (controlled variable).
3. Record at each cycle  $t$ :  $V(t)$ ,  $\alpha(t)$ ,  $S(t)$ , gate satisfaction vector  $(G_A, G_B, G_C, G_D, G_E)$ , finality state, and the set of unresolved contradictions.
4. When finality is reached, continue injecting evidence to trigger re-opening (perpetual lifecycle).
5. Repeat each condition 10 times with different random seeds to estimate variability.

**Expected outcomes.** Three characteristic  $V(t)$  trajectory shapes should emerge: (i) exponential decay when contradictions are low ( $c \leq 1$ ), (ii) plateau-then-resolution when contradictions are moderate ( $c = 3$ ) and require multi-round deliberation, and (iii) escalation when contradiction density is high ( $c = 5$ ) and exceeds autonomous resolution capacity. The re-opening dynamics should show that  $V(t)$  rises sharply on new contradictions and re-converges from the prior graph state rather than from scratch.

**Metrics.** Rounds to first RESOLVED, rounds to re-RESOLVED after re-opening,  $V(t)$  monotonicity violation count, oscillation frequency (Gate C trigger rate), ESCALATED rate.

### 9.2 Experiment 2: Scalability

**Objective.** Provide the first empirical data on how governed agent coordination scales, addressing SECP’s open question on scaling beyond small module sets.

**Protocol.**

1. Vary graph size:  $|\mathcal{N}| \in \{10, 50, 100, 500, 1000\}$  claims.
2. Vary contradiction density:  $\rho \in \{0.10, 0.30, 0.50\}$  (fraction of claim pairs that contradict).
3. Vary agent count:  $|\mathcal{A}| \in \{3, 5, 7, 12\}$ .

4. Fix governance mode to YOLO and finality thresholds to defaults.
5. Measure: rounds to convergence  $k$ , wall-clock time, total LLM token consumption, audit event count, semantic graph query latency.

**Expected outcomes.** Convergence time  $k$  should grow sub-linearly with  $|\mathcal{N}|$  (claims are processed in batches) but linearly or super-linearly with  $\rho$  (each contradiction requires resolution). Pressure-directed activation should demonstrate its value at scale: agents routed to the bottleneck dimension should reduce convergence time compared to round-robin activation. The scaling bottleneck is hypothesized to be contradiction resolution, not claim extraction.

**Metrics.**  $k$  vs.  $|\mathcal{N}|$ ,  $k$  vs.  $\rho$ ,  $k$  vs.  $|\mathcal{A}|$ , LLM tokens per convergence round, pressure-directed activation hit rate (fraction of activations targeting the true bottleneck dimension).

### 9.3 Experiment 3: Finality Robustness Under Adversarial Evidence

**Objective.** Validate that the five-gate mechanism prevents false finality under evidence patterns designed to exploit threshold-based systems.

**Protocol.** Inject four adversarial evidence patterns, each designed to trigger a specific failure mode:

1. **Spike-and-drop:** Inject a batch of high-confidence confirmatory evidence (pushing  $S(t)$  above  $\theta_{\text{auto}}$ ), immediately followed by contradictory evidence in the next cycle.
2. **Oscillating claims:** Alternate between confirmatory and contradictory evidence every cycle for 20 cycles.
3. **Stale evidence:** Allow evidence to age past `max_age_days` during a slow convergence process.
4. **Empty-scope finality:** Initialize a scope with zero claims and zero goals, then add a single high-confidence claim.

**Expected outcomes.** Gate A (monotonicity) should block premature finality in the spike-and-drop scenario by resetting the consecutive-non-decreasing counter. Gate C (oscillation detection) should flag the alternating pattern via negative lag-1 autocorrelation. Gate B (evidence freshness) should block finality on stale data. Gate E (minimum content) should prevent the empty-scope degenerate case.

**Metrics.** False finality rate (RESOLVED with unresolved contradictions), per-gate trigger frequency, trajectory quality score  $Q$  distribution, time in ESCALATED state.

## 9.4 Experiment 4: Multi-Level Governance

**Objective.** Demonstrate governance at multiple bounded levels, extending both this work and SECP’s single-level coordination model.

**Protocol.** Define three governance levels with increasing authority:

- **Level 1 (Operational):** YOLO mode. Handles routine claim extraction and low-drift transitions autonomously.
- **Level 2 (Compliance):** MITL mode. Activated when drift exceeds medium threshold or when financial claims are involved.
- **Level 3 (Regulatory):** MASTER mode. Activated for critical contradictions (e.g., material financial discrepancies, legal liability). Decisions at this level are immutable and cannot be overridden by L1 or L2.

Run the Project Horizon M&A scenario with per-scope governance level overrides: financial claims at L2, patent disputes at L3, technical assessments at L1.

**Expected outcomes.** The majority of decisions (>80%) should be resolved at L1 (deterministic, zero LLM tokens). L2 decisions should cluster around financial contradictions. L3 decisions should be rare but gate-blocking: finality cannot be declared until all L3 issues are resolved by human review. The certificate chain should record the governance level of each decision, enabling per-level audit.

**Metrics.** Decision distribution across levels, escalation frequency, time-to-finality per level, L3 blocking duration, LLM token consumption per level.

## 9.5 Experiment 5: Coverage–Autonomy Trade-off

**Objective.** Empirically map the coverage–autonomy trade-off identified by SECP [3], using governance modes as the control variable.

**Protocol.**

1. Prepare a fixed document corpus (10 documents, ~100 claims, 30 contradictions).
2. Run identical corpus through three governance configurations: YOLO, MITL (with simulated human approvals), and MASTER.
3. Define coverage  $\Delta(\Pi)$  as the number of claims reaching RESOLVED status.
4. Define autonomy  $\Omega(\Pi)$  as the fraction of governance decisions where a single agent’s objection (drift signal) blocked or escalated a proposal.

**Expected outcomes.** YOLO should produce maximum coverage but minimum autonomy (all objections are compensable by governance approval). MASTER should produce minimum coverage but maximum autonomy (any drift signal blocks advancement). MITL should occupy an intermediate position. This directly maps to SECP’s finding: scalar aggregation ( $\sim$ YOLO) achieved coverage of 6/6; hard veto ( $\sim$ MASTER) achieved 0/6; SECP non-scalar ( $\sim$ MITL) achieved 2–3/6.

Additionally, the Lyapunov convergence rate  $\alpha$  should differ characteristically across modes:  $\alpha_{\text{YOLO}} > \alpha_{\text{MITL}} > \alpha_{\text{MASTER}}$ , reflecting the cost of increased evaluator autonomy on convergence speed.

**Metrics.**  $\Delta(\Pi)$  per mode,  $\Omega(\Pi)$  per mode,  $\alpha$  per mode, time-to-finality per mode, human escalation rate (MITL only).

## 10 Enterprise and Regulatory Fitness

Regulated environments—financial services, healthcare, defense, critical infrastructure—impose requirements that go beyond functional correctness. This section maps the architecture’s capabilities to specific regulatory demands, showing that the mechanisms described above are not merely useful but structurally necessary for enterprise deployment.

### 10.1 Temporal Audit for Regulators

Regulatory frameworks frequently require point-in-time reconstruction: demonstrating what was known, when, and what decisions followed. SOX Section 404 requires that internal controls over financial reporting be documented as they existed at the reporting date. IFRS 9 (Expected Credit Loss) requires temporal evidence chains for impairment assessments. Basel III mandates historical risk factor traceability.

The bitemporal semantic graph (Section 4.2) enables these queries directly:

- “**What did the system know on audit date  $T'$ ?**” As-of transaction-time query: filters to rows with `recorded_at`  $\leq T'$  and (`superseded_at`  $> T'$  or not superseded).
- “**What was believed true for reporting period  $T$ ?**” As-of valid-time query: filters to rows with `valid_from`  $\leq T$  and (`valid_to`  $> T$  or open-ended).
- “**What did the system know at  $T'$  about facts valid at  $T$ ?**” Combined bitemporal query: both filters applied simultaneously.

Unlike retrospective log analysis (grepping timestamps in application logs), these are first-class database queries with indexed support, returning structured graph state rather than unstructured log lines.

## 10.2 Cryptographic Non-Repudiation

When a scope reaches finality (Section 5.7), an Ed25519-signed JWS certificate is issued containing: the scope identifier, the finality decision, the timestamp, and content hashes of the governance and finality policy files in effect. This certificate is self-contained: an external party—auditor, counterparty, regulator—can verify the signature and inspect the payload without access to the running system.

The policy-version hashes are critical: they bind the finality decision to the exact rule set that produced it. If governance rules are amended between two finality decisions, the certificates will carry different policy hashes, enabling precise attribution of which decisions were made under which rules.

This addresses requirements in MiFID II (best execution record-keeping), SOX (management sign-off chains), and EMIR (trade repository reporting), where demonstrating *which rules governed a decision* is as important as demonstrating the decision itself.

## 10.3 Policy Change Management

Every `DecisionRecord` (Section 4.3) embeds a `policy_version` computed as a content hash of the governance and finality configuration files. This creates an immutable link between each governance decision and the rule version that produced it, without requiring a separate version-control system.

When an auditor asks “Was this decision made under the pre-amendment or post-amendment compliance rules?”, the answer is a database query on `decision_records.policy_version`. When a policy is updated, all subsequent decisions carry the new hash. Historical decisions retain the old hash. No retrospective rewriting is possible.

This addresses GDPR Article 25 (data protection by design, requiring demonstrable compliance measures), the change control requirements of ISO 27001 Annex A, and the policy versioning demands of PCI-DSS Requirement 6.

## 10.4 Perpetual Compliance: Regulatory Lifecycle Mapping

Most regulated processes are not one-shot: they require ongoing monitoring, periodic reassessment, and event-triggered re-evaluation. The perpetual finality lifecycle (Section 5.8) maps directly to these requirements:

- **KYC/AML ongoing monitoring.** The 5th Anti-Money Laundering Directive requires continuous due diligence—not point-in-time checks. Customer risk profiles must be refreshed annually (standard risk) or quarterly (high risk). Each review cycle ingests new data (transaction patterns, sanctions list updates, adverse media), re-opens convergence, and produces a new finality certificate. The chain of certificates constitutes the monitoring record that regulators inspect.

- **IFRS 9 Expected Credit Loss.** Impairment models must be recalibrated as macroeconomic conditions evolve. Each recalibration is a new convergence cycle over the same graph, with updated economic indicators contradicting or confirming prior forward-looking estimates. The bitemporal graph preserves the temporal evolution of credit risk assessments; the certificate chain traces model updates.
- **Basel III/IV risk-weighted assets.** Quarterly recalculation of capital adequacy ingests new exposure data. Contradictions between prior and current risk factor values trigger drift detection and governance review before the new calculation is finalized. Each quarterly finality certificate binds to the policy rules (including regulatory buffers) in effect at that point.
- **Post-market pharmacovigilance.** After a drug’s marketing authorization (initial finality), adverse event reports continuously feed the context graph. A severe safety signal creates contradictions against prior efficacy claims, driving  $V(t)$  upward and triggering re-convergence. The system may transition to ESCALATED, routing to human review with full context of what changed and why.
- **Sanctions screening.** Daily list updates from OFAC, EU, and UN require re-screening of previously cleared entities. Each update is new context that may contradict prior clearances. A name match creates a contradiction edge; the system re-converges, and a new certificate is issued only when the match is either resolved (false positive) or escalated (true positive requiring action).
- **Post-merger integration.** The M&A close produces one finality certificate. Quarterly integration reports introduce new operational and financial data. The graph absorbs these, drift detection identifies discrepancies from the original due-diligence conclusions, and the system tracks whether pre-close assumptions still hold—producing a running audit trail of integration risk.

In each case, the operational model is the same: a shared context graph that accumulates knowledge over time, a convergence mechanism that produces formal checkpoints, and a certificate chain that provides auditors with a timestamped, policy-versioned, cryptographically signed record of every decision point. No separate “monitoring system” is needed; the same architecture that performs initial analysis performs ongoing surveillance.

## 10.5 Evidence Freshness and Completeness

An evidence schema (Section 4.2) declares, per domain, which evidence types are required and how long each remains valid (`max_age_days`). Gate B of the finality evaluator blocks auto-resolution when:

- A required evidence type is absent from the graph (incomplete coverage).

- Present evidence has exceeded its maximum valid age (stale evidence).

This maps directly to ongoing monitoring requirements in AML regulations (5th Anti-Money Laundering Directive: continuous due diligence, not point-in-time checks), certification freshness in ISO 27001 (controls must be periodically re-validated), and KYC review cycles in financial services (customer risk profiles must be refreshed within regulatory timelines).

## 10.6 Separation of Duties

OpenFGA (Section 4.6) enforces relationship-based access control across four dimensions: document read access, transition proposal rights, approval authority, and policy modification permissions. The three governance modes (YOLO, MITL, MASTER) map to organizational risk appetite:

- **YOLO** for internal, low-sensitivity workflows where speed matters and audit trail suffices.
- **MITL** for standard compliance workflows where every state transition requires human sign-off (SOX Section 302/404 segregation of duties).
- **MASTER** for highest-sensitivity scenarios (e.g., sanctions screening) where LLM non-determinism is unacceptable and only deterministic rules are permitted.

Per-scope overrides enable fine-grained control: within a single M&A due-diligence process, financial claims can require MITL while technical assessments operate in YOLO mode. This granularity is required by PCI-DSS Requirement 6 (role-based access to payment data) and SOX (segregation between preparers and approvers of financial statements).

## 10.7 Operational Resilience

Agent failures do not cascade. When one agent is slow or fails, others continue operating. The disagreement metric  $V(t)$  rises, triggering escalation rules or human routing. The three-node cycle with epoch CAS ensures atomic transitions even under concurrent agent execution.

Three resilience mechanisms ensure continuous governance:

- **Circuit breaker:** LLM failures trigger automatic fallback to Tier 1 deterministic evaluation (3 consecutive failures, 60-second cooldown). Governance never stalls due to LLM unavailability.
- **Message deduplication:** A `processed_messages` table with atomic check-and-mark ensures exactly-once processing of proposals, preventing duplicate governance decisions under message replay.
- **Graceful degradation:** Convergence tracking, embedding pipeline, obligation enforcement, and finality certificate issuance each degrade gracefully when their backing services are unavailable. The system continues at reduced capability rather than halting.

## 10.8 Governance Agility

When business needs change, update the YAML governance rules. The change is auditable (the next decision record carries a new policy-version hash), testable (new rules can be evaluated against the existing event log in a dry-run mode), and live immediately. No code recompilation, no redeployment. The OPA-WASM backend enables enterprise policy teams to author governance rules in Rego using their existing toolchain, compile to WebAssembly, and deploy without modifying agent code.

# 11 The Reasoning Agent Paradigm

The emergence of reasoning-capable LLM agents—systems that can plan, hypothesize, and revise conclusions based on new evidence—fundamentally changes the coordination problem. In the DAG paradigm, agents are execution units: they receive inputs and produce outputs in a predefined sequence. In the reasoning paradigm, agents are *epistemic actors*: they maintain beliefs, generate hypotheses, and update them as evidence accumulates.

This shift has three implications for coordination:

**Contradiction is information, not failure.** When two reasoning agents disagree about a fact (e.g., conflicting revenue figures from different documents), this disagreement is valuable signal. A governance-based system can track contradictions explicitly in the semantic graph, route them to investigation, and declare finality only when they are resolved—or escalated to human judgment with full context.

**Coordination must be emergent, not hardcoded.** Reasoning agents discover new information that may invalidate the planned execution path. A governance-based system allows agents to propose new directions based on what they find, evaluated by declarative rules rather than pre-determined sequences. The three-node cycle (ContextIngested → FactsExtracted → DriftChecked) provides structure without rigidity.

**Finality requires formal guarantees.** When agents reason and revise, the system may never converge without formal mechanisms. Lyapunov-based convergence tracking provides mathematical guarantees that the system either reaches consensus or explicitly routes to human review—it cannot cycle indefinitely.

Our system is designed for this paradigm: agents reason independently over shared context, governance rules coordinate their outputs, and formal mechanisms ensure convergence. As LLM agents become more capable (chain-of-thought reasoning, tool use, multi-step planning), the governance-based coordination model becomes more relevant, not less.

# 12 Scope and Boundaries

Following the principled scoping methodology of de la Chica Rodriguez and Vera Díaz [3], we state explicitly and without mitigation rhetoric what this work establishes and what it does not.

## 12.1 What the Paper Demonstrates

1. **Architectural feasibility.** Declarative governance over shared immutable context is implementable with current technology (Postgres, NATS, LLMs, OpenFGA) and produces a functioning coordination system with 197 unit tests and 7 convergence benchmarks.
2. **Formal convergence tracking.** The Lyapunov disagreement function  $V(t)$  with five independent gates prevents premature finality in all tested scenarios (steady convergence, plateau, spike-and-drop, divergence, one-dimension bottleneck, fast convergence, empty graph). Proposition 1 provides a proof sketch for monotonic progress under CRDT constraints.
3. **Perpetual lifecycle.** The architecture supports re-opening finality on new evidence without information loss, producing a chain of signed certificates that models the perpetual nature of regulated knowledge.
4. **Three-tier governance resilience.** The system continues to make governance decisions without LLM availability, falling back to deterministic rule evaluation via the circuit-breaker mechanism.
5. **End-to-end validation.** The Project Horizon scenario demonstrates 83% autonomous contradiction resolution across 5 contradictory documents in 12 convergence rounds, with complete audit trail.
6. **Coverage-autonomy trade-off.** The three governance modes (YOLO, MITL, MASTER) provide distinct positions on the coverage-autonomy spectrum identified by [3]: YOLO maximizes coverage (analogous to scalar aggregation), MASTER maximizes evaluator autonomy (analogous to hard veto), and MITL provides an intermediate, auditable regime.

## 12.2 What the Paper Does Not Demonstrate

1. **Statistical generality or significance.** Project Horizon is a single scenario with synthetic documents. No confidence intervals, hypothesis tests, or distributional claims are supported. Results are existence proofs, not performance estimates.
2. **Byzantine fault tolerance.** All agents are assumed cooperative. The system provides no defense against adversarial agents that strategically inject false claims, inflate confidence scores, or collude to manipulate governance decisions. The monotonic confidence constraint limits downward manipulation but does not prevent upward inflation. Byzantine-tolerant coordination [7, 2, 19] remains future work.
3. **Scalability beyond proof of concept.** Validation covers 50 claims, 5 documents, and 7 agents. Whether convergence dynamics, governance throughput, or the Lyapunov function's behavior change qualitatively at production scale (thousands of claims, hundreds of agents) is unknown.

4. **Machine-checked convergence proofs.** Proposition 1 is a proof sketch validated empirically. No formal verification tool or proof assistant has checked the convergence guarantee. An arbitrary sequence of approved transitions could, in principle, violate the proposition’s preconditions in untested scenarios.
5. **Convergence under real LLM stochasticity.** All convergence benchmarks use synthetic trajectories. Whether the oscillation detection (Gate C) and trajectory quality score  $Q$  perform correctly under the inherent stochasticity of real LLM reasoning traces is untested.
6. **Multi-iteration protocol self-modification.** Governance rules are declarative but static within a deployment. The system does not implement governed self-modification of its own coordination rules, as demonstrated by SECP [3]. Whether the Lyapunov convergence mechanism can serve as a formal foundation for evaluating protocol modifications across multiple iterations is a promising but unvalidated direction.
7. **Multi-scope and hierarchical finality.** Cross-scope governance, parent–child finality dependencies, and inter-scope certificate chains are designed in the architecture but not validated experimentally.
8. **Human-in-the-loop effectiveness.** The HITL review mechanism is designed and implemented but not empirically evaluated with real human reviewers.
9. **Coverage–autonomy formalization.** While the three governance modes map conceptually to the coverage–autonomy trade-off, no formal metric quantifies evaluator autonomy in our system, and no systematic experiment measures the trade-off across modes on identical inputs.

Any claim beyond the narrow statement—“the architecture is implementable, produces auditable convergence in a controlled scenario, and provides formal safeguards against premature finality”—is unsupported by the evidence presented.

## 13 Limitations and Future Work

This section states the principal limitations and the research directions required to move from architectural feasibility to practical validation.

### 13.1 Implementation Limitations

**Dual temporality utilization.** The bitemporal schema is structurally complete (valid-time and transaction-time columns, as-of queries, temporal contradiction detection). However, the current facts extraction pipeline does not systematically populate `valid_from`/`valid_to` from document metadata. Most nodes are created without explicit valid-time, treated as open-ended. Extracting temporal bounds from unstructured documents (e.g., “FY2024 revenue” → `valid_from: 2024-01-01, valid_to: 2025-01-01`) requires temporal NLP or structured metadata ingestion.

Until this is implemented, dual temporality is architecturally present but operationally underutilized.

**Evidence schemas.** The evidence schema mechanism (Gate B) is framework-ready but currently configured with empty required types and no staleness constraints. Domain-specific schemas (e.g., M&A due diligence requiring financial, legal, technical, and compliance evidence types with maximum age constraints) must be authored per deployment.

**Obligation handlers.** The obligation enforcement framework provides a registry-based architecture and execution pipeline, but no handlers are currently registered. Real deployments will need handlers for dual review, compliance notification, escalation to external systems, and audit-entry generation.

## 13.2 Scalability Limitations

**Horizontal scaling.** The current implementation operates on a single Postgres instance, a single NATS stream, and a single governance agent process. While the architecture (event-driven, scope-partitioned, stateless agents) is designed for horizontal scaling, no sharding strategy, multi-instance governance, or distributed CAS has been validated. Production deployments at enterprise scale will require these.

**Graph scale.** Validation at 50 claims with 30% logical overlap. Production scale (thousands of claims, hundreds of agents) requires semantic graph optimization (approximate nearest-neighbor search, event log sharding, compiled governance rules).

## 13.3 Theoretical Limitations

**Convergence proof gap.** Proposition 1 provides a proof sketch, not a machine-checked proof. The proposition’s preconditions (every approved transition satisfies monotonicity constraints and makes progress on at least one dimension) are enforced by the system’s design but not formally verified. A rigorous proof would require formalizing the governance function, the CRDT semantics, and the gate predicates in a proof assistant such as Coq or Lean.

**Scalar convergence metric.** The Lyapunov disagreement function  $V(t)$  is a single scalar aggregating four weighted dimensions. This mirrors the scalar aggregation that SECP [3] criticizes for collapsing structured disagreement. A multi-dimensional convergence criterion—where finality requires per-dimension satisfaction rather than aggregate threshold crossing—would provide richer guarantees but complicates the mathematical analysis.

**Static governance rules.** The governance function  $\Pi$  (Definition 3) is fixed within a deployment. Unlike SECP’s self-modifying coordination protocols, our system does not adapt its governance rules based on observed outcomes. Integrating governed protocol modification [3] with Lyapunov convergence tracking is a natural extension: each protocol modification would be evaluated by its effect on  $V(t)$ , with gates preventing modifications that increase disagreement.

**Coverage–autonomy trade-off.** The coverage–autonomy trade-off identified by [3] manifests in our three governance modes but is not formalized. Defining evaluator autonomy as a measurable

quantity (e.g., the probability that a single agent’s objection blocks finality) and mapping it against coverage across governance configurations would provide a principled basis for mode selection.

### 13.4 Future Work: Research Directions

1. **Multi-iteration convergence experiments.** Run repeated convergence cycles with incremental evidence injection to characterize  $V(t)$  trajectory shapes, convergence time distributions, and gate activation patterns (Section 9).
2. **Scalability benchmarks.** Systematically vary graph size, contradiction density, and agent count to identify scaling bottlenecks and validate pressure-directed activation at scale.
3. **Byzantine agent integration.** Extend the cooperative-agent model with Byzantine fault tolerance, potentially adopting confidence-weighted consensus [19] or adapting the SECP non-scalar coordination mechanism to handle adversarial module outputs.
4. **Governed protocol evolution.** Integrate SECP-style bounded self-modification [3] with Lyapunov convergence tracking, using  $V(t)$  as the objective function for protocol modification and gates as invariant-preservation checks.
5. **Multi-scope hierarchical finality.** Validate cross-scope governance where a parent scope reaches finality only when all child scopes are resolved, with certificate chains reflecting the hierarchy.
6. **Formal verification.** Encode the convergence proposition, gate predicates, and governance invariants in a proof assistant to obtain machine-checked guarantees.
7. **Real LLM trajectory validation.** Replace synthetic benchmarks with recorded LLM reasoning traces to validate oscillation detection and trajectory quality mechanisms under real stochasticity.
8. **Human-in-the-loop evaluation.** Conduct user studies with domain experts to measure HITL review effectiveness, decision quality, and time-to-resolution.

## 14 Conclusion

### 14.1 Summary of Contributions

This paper presents a paradigm for autonomous agent coordination based on declarative governance over shared immutable context. The core contributions are: (i) a formal convergence mechanism (Lyapunov disagreement function with five independent gates) that provides layered guarantees against premature finality; (ii) a bitemporal CRDT-inspired semantic graph with monotonicity invariants ensuring that shared state evolves toward resolution; (iii) a perpetual finality lifecycle producing chains of Ed25519-signed certificates; (iv) a three-tier governance routing architecture

ensuring continuous operation without LLM availability; and (v) validation on a realistic M&A due-diligence scenario.

## 14.2 Positioning Relative to Self-Evolving Coordination

This work and the Self-Evolving Coordination Protocol (SECP) study [3] address complementary halves of a shared problem. Where SECP demonstrates that bounded, governed self-modification of coordination protocols is technically feasible in a single-shot design, we demonstrate that formal convergence tracking over shared state provides the multi-iteration foundation such mechanisms need for sustained deployment. Specifically:

- SECP’s open question on multi-iteration dynamics is addressed by our Lyapunov function  $V(t)$ , which tracks convergence over indefinite cycles with formal safeguards against oscillation and premature finality.
- SECP’s open question on audit infrastructure is addressed by our bitemporal graph and policy-version-bound certificate chain, which provide the temporal auditability their architecture currently checks empirically.
- Our open question on non-scalar coordination is addressed by SECP’s Pareto filtering, minimax-regret screening, and constructive-objection requirements, which prevent the collapse of structured disagreement that our scalar  $V(t)$  metric permits.
- Our open question on adversarial agents is partially addressed by SECP’s Byzantine fault tolerance framing and their non-compensable objection rights.

A synthesis—governed protocol evolution evaluated by Lyapunov convergence, with non-scalar coordination within each cycle and formal finality gates across cycles—would address the limitations of both approaches. The proposed experimental protocol (Section 9) is designed to build toward this synthesis.

## 14.3 Interpretation Boundaries

The empirical claims are narrow and intentionally circumscribed (Section 12). Results are existence proofs from a single validation scenario, not distributional claims. The convergence guarantee (Proposition 1) is a proof sketch, not a machine-checked theorem. No adversarial testing, no real human-in-the-loop evaluation, and no production-scale validation have been performed. Any interpretation beyond “the architecture is implementable and produces auditable convergence in a controlled scenario” exceeds the evidence presented.

## 14.4 Final Assessment

As LLM agents become increasingly capable of reasoning, planning, and hypothesis generation, the need for governance-based coordination—rather than pipeline-based orchestration—will grow.

Regulated enterprises cannot adopt autonomous agents without the ability to audit, explain, and formally bound their behavior, not once but continuously as conditions evolve. The mechanisms that enable governance—shared state, declarative policy, formal convergence—are precisely the mechanisms that make multi-agent coordination robust over indefinite operational horizons. Achieving this robustness requires combining the state-level convergence approach presented here with the protocol-level coordination adaptability demonstrated by SECP. Neither alone is sufficient; together, they define the research agenda for governed autonomous agent systems in regulated environments.

## Acknowledgments

This work was supported by Deal ex Machina SAS. We thank the communities behind PostgreSQL, NATS, OpenFGA, DSPy, and Mastra for foundational infrastructure.

## References

- [1] Alberto Camacho et al. MACI: EMA-based stagnation detection in multi-agent coordination. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.
- [2] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.
- [3] Jose Manuel de la Chica Rodriguez and Juan Manuel Vera Díaz. Self-evolving coordination protocol in multi-agent AI systems: An exploratory systems feasibility study. *arXiv preprint arXiv:2602.02170*, 2026.
- [4] Marco Dorigo et al. Swarm intelligence: From stigmergic coordination to pressure-directed activation. *Swarm Intelligence*, 18(1):1–28, 2024.
- [5] Haibin Duan et al. Aegean: Monotonicity gates and coordination invariants for multi-agent convergence. 2025. Preprint.
- [6] Shadaj Laddad et al. CodeCRDT: Monotonic merge operations for distributed collaborative editing. In *Proceedings of the ACM on Programming Languages (OOPSLA)*, 2024.
- [7] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
- [8] LangChain Team. LangGraph: Building stateful, multi-actor applications with LLMs. <https://github.com/langchain-ai/langgraph>, 2024.

- [9] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *Annals of the Faculty of Sciences of Toulouse*, 1892. Reprinted in English by International Journal of Control, 1992.
- [10] Yiming Mao, Yao Kang, Peng Li, Nan Zhang, Wei Xu, and Chongjie Zhang. IBGP: Imperfect Byzantine generals problem for zero-shot robustness in communicative multi-agent systems. 2024. Preprint.
- [11] João Moura. CrewAI: Framework for orchestrating role-playing autonomous AI agents. <https://github.com/joaomdmoura/crewAI>, 2024.
- [12] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0. Technical report, OASIS Standard, January 2013. <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>.
- [13] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.
- [14] Open Policy Agent Contributors. Open Policy Agent: Policy-based control for cloud native environments. <https://www.openpolicyagent.org/>, 2021. CNCF Graduated Project.
- [15] Ruoming Pang et al. Zanzibar: Google’s consistent, global authorization system. In *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC)*, pages 33–46, 2019.
- [16] Wei Ren, Randal W. Beard, and Ella M. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005 American Control Conference*, pages 1859–1864. IEEE, 2005.
- [17] Richard T. Snodgrass. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 2000. Definitive reference on bitemporal database modeling.
- [18] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- [19] Luyao Zheng, Jianhua Chen, Quanjun Yin, Junping Zhang, Xingguo Zeng, and Yonghong Tian. Rethinking the reliability of multi-agent system: A perspective from Byzantine fault tolerance. 2025. Preprint.