CONTRACT-GOVERNED MULTI-AGENT GRAPH ORCHESTRATION FOR LONG-HORIZON AUTONOMOUS RESEARCH PIPELINES

Anonymous authors
Paper under review

ABSTRACT

Long-horizon autonomous research systems now execute multi-stage workflows with planner, tool, and reviewer modules, yet most guarantees remain local or empirical. We formalize a contract-governed orchestration model in which execution is a finite directed acyclic phase graph with typed interfaces, adapter maps, decidable transition guards, and validator-mediated control. Within this setting, we define a safety objective that minimizes contract violations and a recoverability objective that minimizes halting time under bounded control. We prove a graph-level safety theorem by composing edge admissibility and invariant propagation, and we prove a bounded recoverability theorem with an explicit finite halt envelope; we also prove an impossibility boundary when boundedness or monotone escalation is removed. The formal statements are paired with theorem-audit evidence from symbolic checks and seeded trace sweeps: fail-closed bounded runs satisfy the proven envelope, while fail-open or non-monotone regimes produce counterexamples exactly where the theory predicts failure. The resulting manuscript gives a theorem-first blueprint for contract-compositional orchestration that separates theorem-supported claims from diagnostic, out-of-regime observations.

1 INTRODUCTION

Autonomous research pipelines increasingly combine planning, retrieval, coding, simulation, and writing in one execution loop. Multi-agent frameworks have improved practical throughput by decomposing tasks across specialized roles and by maintaining structured interaction traces (Song et al., 2026; Wu et al., 2023; Hong et al., 2023; Qian et al., 2023; Li et al., 2023; Chen et al., 2023; Liu et al., 2023b). At the same time, benchmark advances in software and web environments have shown that long-horizon systems are fragile under interface mismatch, weak verification gates, and ambiguous control policies (Liu et al., 2023a; Zhou et al., 2023; Jimenez et al., 2023; Yang et al., 2024; Wang et al., 2024). These observations motivate a formal question: under what conditions can a multi-agent orchestration graph guarantee safety of phase contracts and bounded recoverability after validation failures?

This paper answers that question for a proof-first regime aligned with a concrete orchestration stack. The stack contains a phase graph, typed phase contracts, coordinator policies, and validator outcomes. We formalize those components as mathematical objects, then derive and prove two central guarantees. The first guarantee is global contract safety under typed-adapter closure, decidable guards, and fail-closed validation. The second guarantee is bounded recoverability under bounded branching, bounded backtracking, and monotone escalation with loop detection. We then characterize the boundary where guarantees fail: fail-open validation for safety, and unbounded or non-monotone control for recoverability. This decomposition lets us keep positive guarantees precise while preserving negative evidence instead of averaging it away.

The work matters beyond one implementation because modern orchestrators face the same structural tension: practical systems require adaptivity, but formal guarantees require explicit regimes. Classical program logic and contract systems provide local correctness tools (Meyer, 1992; Jazequel & Meyer, 1997; Findler & Felleisen, 2002; Hoare, 1969), timed and temporal verification provide finite-state reasoning patterns (Pnueli, 1977; Alur & Dill, 1994; Clarke et al., 1986; de Moura & Bjørner, 2008; , editor), and empirical autonomous-research systems provide operational context (Lu et al., 2024; 2026). Our contribution is to connect these strands in a single theorem-bearing model tuned to graph-structured research pipelines.

Methodologically, we also treat manuscript traceability as part of the scientific object. Every theorem-level statement is paired with a regime qualifier and an explicit evidence anchor so that readers can distinguish constructive guarantees from contradiction-boundary demonstrations. This framing is important for deployment-facing research pipelines,

where safety claims are often reused outside their validity conditions. By making the regime and evidence links first-class, we reduce the risk of overgeneralized interpretation while preserving reproducibility for later extension.

Contributions.

- We define a contract-governed phase-graph model with typed spaces, adapter maps, decidable guards, and validator semantics that explicitly marks in-regime and out-of-regime behavior.
- We prove a contract-compositional safety theorem by combining edge admissibility and invariant propagation, and we provide a complete counterexample construction showing sharpness when adapter closure is violated.
- We prove a bounded recoverability theorem with an explicit halt envelope H and a complete impossibility theorem outside bounded monotone control.
- We connect theorem statements to symbolic and seeded validation artifacts through a claim-evidence map that preserves conditional support rather than forcing scalarized pass/fail conclusions.

1.1 Problem Importance and Scope

Long-horizon orchestration is not only an engineering convenience; it is a reliability bottleneck for any automated research service that must maintain validity across multiple decision epochs. Without typed phase interfaces, downstream modules can consume malformed state. Without explicit validator semantics, reported quality can conflate theorem-confirming and out-of-regime traces. Without bounded control assumptions, recoverability claims can silently rely on finite traces that are not guaranteed to terminate. We therefore focus on a mathematically auditable scope: finite directed acyclic phase graphs, typed phase contracts, explicit control actions, and solver-checkable guards. This scope excludes unconstrained natural-language transition predicates and fully open-ended branch creation. The exclusions are deliberate because they preserve theorem interpretability while still matching practical orchestrator primitives seen in agentic systems (Song et al., 2026; Dinu et al., 2024; Lu et al., 2024; Yang et al., 2024; Lu et al., 2026).

2 Related Work and Novelty Boundary

This section organizes prior work by the exact technical roles it plays in the present manuscript: empirical motivation, formal foundations, and contradiction boundaries that determine what can and cannot be claimed. We use this structure to avoid a citation-list treatment. Instead of describing papers independently, we align them against three questions: how they model orchestration state, how they encode correctness obligations, and how they report evidence under changing regimes. The synthesis exposes why theorem-bearing claims need explicit interface and control assumptions even when empirical results are strong. It also clarifies where our contribution is incremental reuse versus manuscript-specific formalization choices.

2.1 Multi-Agent Orchestration and Evaluation

Recent frameworks establish that role decomposition improves long-horizon execution in complex tasks, including software workflows, writing systems, and autonomous experimentation (Song et al., 2026; Wu et al., 2023; Hong et al., 2023; Qian et al., 2023; Li et al., 2023; Chen et al., 2023; Liu et al., 2023b; Lu et al., 2024; 2026). Benchmarks such as AgentBench, WebArena, and SWE-bench sharpen operational evaluation by exposing environment-grounded failure modes (Liu et al., 2023a; Zhou et al., 2023; Jimenez et al., 2023; Yang et al., 2024). Dynamic benchmark proposals further argue that static splits may hide temporal drift in measured capability (Wang et al., 2024). The strength of this literature is empirical breadth and practical realism. Its limitation for theorem-bearing claims is that correctness assumptions are often implicit: interfaces, guards, and escalation rules are frequently represented as prompts or heuristics rather than machine-checkable predicates. Our manuscript builds on this literature for motivation and regime diagnostics, but theorem claims are restricted to the explicit formal setting in section 3. Complementary lines of work on deliberate reasoning and tool-use policies further show that trajectory quality depends on control-logic structure, not only base-model capability (Yao et al., 2023; Wang et al., 2023a;b; Shinn et al., 2023; Schick et al., 2023; Yao et al., 2022; Madaan et al., 2023; Khattab et al., 2023). Multi-agent design studies also report that communication topology, memory organization, and role protocol choices influence reliability under long-horizon workloads (Qian et al., 2024; Ferrag et al., 2025; Händler, 2023). These observations reinforce our decision to formalize policy and validator semantics as first-class mathematical objects rather than implementation afterthoughts.

2.2 Contracts, Program Logic, and Transition Verification

Design-by-contract and Hoare-logic traditions provide compositional language for preconditions, postconditions, and local correctness (Meyer, 1992; Jazequel & Meyer, 1997; Findler & Felleisen, 2002; Hoare, 1969). Temporal and timed formalisms provide analyzable transition semantics and bounded reasoning mechanisms (Lamport, 1978; Pnueli, 1977; Alur & Dill, 1994; Clarke et al., 1986). SMT-based workflows provide decidable satisfiability engines suitable for guard-level checks in finite encodings (de Moura & Bjørner, 2008). Runtime verification adds monitor semantics for online acceptance and rejection under observed traces (, editor). Distributed-snapshot and workflow-net perspectives add useful abstractions for persistent-state consistency, process decomposition, and orchestration/choreography boundaries (Chandy & Lamport, 1985; Rajsbaum, 2001; AALST, 1998; Peltz, 2003; Camilli, 2020; van der Aalst, 2013). The strength of this formal lineage is precision: assumptions and validity regimes can be written explicitly. The limitation is representational mismatch with adaptive, role-based LLM orchestration unless one specifies bounded policy spaces and typed adapter maps. Our contribution occupies this gap by adapting formal ingredients to a graph-governed multi-agent controller and proving both positive and negative boundary statements in that setting.

2.3 Novelty Boundary and Claim Discipline

Our novelty is not a general theorem for all autonomous-agent systems; it is a theorem family for a specific but practically meaningful regime: finite DAG orchestration, typed adapters, decidable guards, fail-closed validation, and bounded monotone recovery control. Within this regime, we prove global safety and bounded recoverability. Outside this regime, we explicitly prove impossibility for finite halt bounds and report safety degradation as boundary diagnostics. This claim discipline avoids two common errors: overextending empirical success into universal guarantees, and overgeneralizing finite-state theorems to open-ended policies. We maintain a three-tier evidence interpretation throughout the manuscript: theorem-supported, mixed at boundary, and pending due to external-provenance limits.

3 Problem Formulation

The mathematical setting is chosen to preserve both practical relevance and proof tractability. The model captures graph-structured orchestration with typed interfaces and validator feedback, which are common in long-horizon systems, while excluding open-ended control semantics that would make theorem interpretation ambiguous. We present objects and assumptions in the order they are consumed by later proofs: first phase and edge objects, then policy and objectives, then regime partitions. This ordering ensures every theorem premise in later sections has a first-use definition with domain and applicability constraints already stated.

3.1 Objects, Spaces, and Interfaces

Let $G = (V, E)$ be a finite directed acyclic graph with node set $V = \{1, \dots, N\}$ and $N \in \mathbb{N}_{\geq 1}$. Each node $v \in V$ denotes a phase operator with typed input space X_v and typed output space Y_v, both measurable spaces. A phase map is a total function on its admissible input set: $F_v : P_v \to Q_v$, where $P_v \subseteq X_v$ and $Q_v \subseteq Y_v$. For each edge $(u, v) \in E$, an adapter map $\kappa_{uv} : Q_u \to X_v$ translates outputs of phase u into inputs for phase v. Transition guards are formulas ϕ_{uv} in a decidable logic fragment $\mathcal{L}_{\mathrm{dec}}$. The admissibility predicate for edge (u, v) is

$$\mathcal{A}_{uv}(y_u) := \big(\mathrm{Sat}(\phi_{uv}) = 1\big) \wedge \big(\kappa_{uv}(y_u) \in P_v\big). \tag{1}$$

The first term is borrowed from solver-based transition checks (Alur & Dill, 1994; de Moura & Bjørner, 2008); the second term is a contract-closure condition adapted from interface-correctness traditions (Meyer, 1992; Jazequel & Meyer, 1997; Hoare, 1969).

Execution state at discrete time $t \in \mathbb{N}$ is $z_t = (v_t, x_t, m_t, h_t)$, where $v_t \in V$ is current phase, $x_t \in X_{v_t}$ is payload, m_t is finite metadata, and h_t is finite trace history. The controller action alphabet is

$$\mathcal{A} = \{\mathrm{continue}, \mathrm{reroute}, \mathrm{backtrack}, \mathrm{halt}\}. \tag{2}$$

A policy $\pi : \mathcal{Z} \to \mathcal{A}$ maps current state to one control action. A validator family $\mathcal{V} = \{\nu_i\}_{i=1}^{k}$ returns discrete outcomes in $\{\mathrm{pass}, \mathrm{fail}, \mathrm{abstain}\}$ and is aggregated by rule Γ. In this paper's theorem regime, fail-closed semantics means any hard fail blocks downstream write and triggers halting or controlled backtracking.

3.2 Objective, Feasible Set, and Optimality Criteria

We formalize two objectives because the selected contribution is hybrid safety-progress mathematics. The safety objective penalizes contract violations along a run of length T:

$$J_{\text{safe}}(\pi) := \sum_{t=0}^{T-1} \mathbf{1}\{x_t \notin P_{v_t} \ \vee \ y_t \notin Q_{v_t} \ \vee \ \mathcal{A}_{v_t v_{t+1}}(y_t) = 0\}. \tag{3}$$

Here $y_t = F_{v_t}(x_t)$ whenever $x_t \in P_{v_t}$. The optimization target for safety is

$$\pi^\star_{\text{safe}} \in \arg\min_{\pi \in \Pi_{\text{reg}}} J_{\text{safe}}(\pi), \tag{4}$$

where Π_{reg} is the theorem-regime policy class defined below.

For recoverability, define halt time $\tau_{\text{halt}}(\pi)$ as first t with action halt. Under bounded branching factor $b \in \mathbb{N}_{\geq 1}$ and backtrack depth $d \in \mathbb{N}_{\geq 0}$, define finite envelope

$$H := N \sum_{i=0}^{d} b^i. \tag{5}$$

The recoverability objective is to minimize halt time subject to safety feasibility:

$$\pi^\star_{\text{rec}} \in \arg\min_{\pi \in \Pi_{\text{reg}}} \tau_{\text{halt}}(\pi) \quad \text{s.t.} \quad J_{\text{safe}}(\pi) = 0. \tag{6}$$

The feasible set Π_{reg} consists of policies satisfying seven standing assumptions: finite DAG execution, typed total phase maps on contract domains, adapter closure on traversed edges, decidable guards, fail-closed validator aggregation, admissible initial state, and explicit exclusion of opaque guard languages. These assumptions are inherited from earlier formal-method lineages (Hoare, 1969; Alur & Dill, 1994; Clarke et al., 1986; de Moura & Bjørner, 2008; , editor) and adapted to orchestrated research pipelines (Song et al., 2026; Dinu et al., 2024; Lu et al., 2024; 2026).

3.3 Validity Regimes and Exclusions

Theorem statements in section 4 and section 5 apply only to *strict theorem regime* runs: fail-closed validators, decidable guards, bounded branching and depth, and monotone escalation. A *relaxed regime* allows partial relaxations but cannot upgrade theorem claims without re-proof. An *out-of-regime* run violates one or more theorem assumptions; these runs are informative diagnostics but not positive confirmation. This separation is central to claim discipline because the same architecture can host both theorem-valid and theorem-invalid trajectories. In later sections we report both kinds of evidence and keep their interpretive status explicit.

4 Contract-Compositional Safety Analysis

The goal of this section is to lift local contract checks to a global trajectory guarantee without introducing hidden assumptions. We proceed from definitions to lemmas and then to the main theorem so each inferential dependency is visible. Local lemmas encode edge-level preservation and one-step propagation. The main theorem then composes these lemmas over execution horizon with fail-closed transition semantics. We also include a sharpness subsection because positive guarantees are incomplete without explicit failure constructions at assumption boundaries. This proof ordering mirrors how a reviewer would audit mechanized obligations: local admissibility first, then composition, then failure witnesses.

4.1 Definitions and Local Lemmas

Define the global invariant I_t by

$$I_t := \big(x_t \in P_{v_t}\big) \wedge \big(y_t = F_{v_t}(x_t) \in Q_{v_t}\big). \tag{7}$$

The invariant encodes contract satisfaction at step t. It uses local contract objects borrowed from design-by-contract and Hoare-style reasoning, but the time-indexed graph coupling is defined in this work for orchestration trajectories.

Lemma 4.1 (Edge Preservation)**.** *Assume I_t holds, and transition $(v_t, v_{t+1}) \in E$ is taken with $\mathcal{A}_{v_t v_{t+1}}(y_t) = 1$. Then $x_{t+1} = \kappa_{v_t v_{t+1}}(y_t) \in P_{v_{t+1}}$.*

Proof. From I_t we have $y_t \in Q_{v_t}$. Because $\mathcal{A}_{v_t v_{t+1}}(y_t) = 1$, by equation 1 we have both satisfiable guard and $\kappa_{v_t v_{t+1}}(y_t) \in P_{v_{t+1}}$. By transition semantics, the next input equals that adapter image, hence $x_{t+1} \in P_{v_{t+1}}$. □

Lemma 4.2 (One-Step Invariant Propagation)**.** *Under the assumptions of Lemma 4.1 and totality of $F_{v_{t+1}}$ on $P_{v_{t+1}}$, I_{t+1} holds.*

Proof. Lemma 4.1 gives $x_{t+1} \in P_{v_{t+1}}$. Totality of $F_{v_{t+1}}$ on $P_{v_{t+1}}$ implies $y_{t+1} = F_{v_{t+1}}(x_{t+1})$ is defined and belongs to $Q_{v_{t+1}}$ by codomain contract. Therefore both conjuncts in equation 7 hold at time $t+1$. □

4.2 Main Safety Theorem

Theorem 4.3 (Global Contract Safety)**.** *Suppose execution starts from admissible $x_0 \in P_{v_0}$, guards are checked in $\mathcal{L}_{\text{dec}}$, adapters satisfy closure on every traversed edge, and validator aggregation is fail-closed. Then for every theorem-regime trajectory and every $t \le T$, invariant I_t holds. Consequently,*

$$J_{\text{safe}}(\pi) = 0 \tag{8}$$

for all $\pi \in \Pi_{\text{reg}}$, so the optimal value in equation 4 equals zero.

Proof. Base case: admissible initialization gives $x_0 \in P_{v_0}$, and totality of F_{v_0} yields $y_0 \in Q_{v_0}$, so I_0 holds. Inductive step: assume I_t. In fail-closed mode, only admissible edges may advance state; otherwise execution halts or backtracks without new downstream state. For an advancing edge, admissibility implies Lemma 4.1; then Lemma 4.2 gives I_{t+1}. Hence I_t holds for all reachable t by induction. Because each summand in equation 3 is the indicator of invariant failure or inadmissible transition, every summand is zero on theorem-regime trajectories, proving equation 8. The optimizer value in equation 4 is therefore zero. □

Corollary 4.3.1 (Compositional Safety Certificate)**.** *If edge-level admissibility certificates are available for all traversed edges, the run-level contract-safety certificate follows without additional global solver calls.*

Proof. The theorem proof uses only local admissibility and totality assumptions in each inductive step. Thus global safety is a compositional consequence of local certificates. □

4.3 Sharpness: Counterexample Class

Theorem 4.3 is sharp with respect to adapter closure. Consider scalar predicates q (source postcondition witness) and p (target precondition witness), with implication $q \le 0 \Rightarrow p \le 0$ encoding a simplified closure condition. The symbolic witness $q = -1$, $p = 2$ satisfies $q \le 0$ but violates $p \le 0$, so closure fails and invariant propagation is blocked. This is not a pathological proof artifact: it is exactly the operational failure mode in which a syntactically valid output is semantically inadmissible for the downstream phase. In section 7, the corresponding ablation shows reduced invariant preservation under fail-open semantics, matching this formal boundary.

5 Bounded Recoverability and Impossibility Boundary

Safety alone is not sufficient for long-running orchestrators: systems must also halt or recover within predictable limits after failures. This section gives a theorem pair that separates positive bounded guarantees from negative impossibility boundaries. The positive side establishes a finite halt envelope under bounded monotone control. The negative side proves that removing those restrictions destroys any uniform finite bound. Presenting both sides in the same formal language prevents accidental overextension of bounded results to unbounded regimes. By coupling these statements, we make boundary behavior a theorem object rather than a post-hoc exception note.

5.1 Bounded Tuple-Space Lemma

Define tuple state $\xi = (v, c, \ell)$ where $v \in V$, c is contract hash, and $\ell \in \{0, \dots, d\}$ is escalation depth. Under bounded branching b and depth d, the number of reachable tuples is upper bounded by H from equation 5. The tuple abstraction is crucial because recoverability depends on control-state exhaustion, not only on graph size. Two runs on the same graph can have very different halting behavior if escalation semantics differ. By indexing tuples with escalation depth and contract hash, we preserve enough state to reason about revisits and loop detection while remaining finite under bounded assumptions. This construction also aligns with practical coordinator implementations where replay detection uses compact state fingerprints rather than full execution traces.

Lemma 5.1 (Finite Reachable Tuple Space)**.** *Under finite N, bounded b, and bounded d, the reachable tuple set has cardinality at most $H = N \sum_{i=0}^{d} b^i$.*

Proof. At escalation level i, each node can branch to at most b^i descendants in the policy tree, and there are N nodes. Therefore level-i tuple count is bounded by Nb^i. Summing over $i = 0, \ldots, d$ yields at most $N \sum_{i=0}^{d} b^i = H$ tuples. □

5.2 Recoverability Theorem and Impossibility Theorem

Theorem 5.2 (Bounded Recoverability)**.** *Assume the safety conditions of Theorem 4.3 hold on executed transitions, loop detection forbids revisiting an identical tuple after escalation budget is exhausted, and escalation is monotone. Then every theorem-regime trajectory halts in at most H steps:*

$$\tau_{\text{halt}}(\pi) \leq H. \tag{9}$$

Hence any feasible minimizer in equation 6 exists over a finite horizon.

Proof. By Lemma 5.1, at most H distinct tuples are reachable. Monotone escalation and loop detection prevent infinite cycling over previously exhausted tuples. Therefore each step either consumes a previously unseen reachable tuple or halts/backtracks within remaining budget. After at most H tuple consumptions, no unseen reachable tuple remains, so continuation is impossible and halt must occur. This gives equation 9. □

Theorem 5.3 (Impossibility Outside Bounded Monotone Regime)**.** *If either (i) branching is unbounded or (ii) escalation is non-monotone with reset-enabled revisits, then no uniform finite constant $\bar{H}$ can satisfy $\tau_{\text{halt}}(\pi) \leq \bar{H}$ for all policies and initial states.*

Proof. Case (i): with unbounded branching, construct a policy that emits a fresh branch index at each step while avoiding halt; tuple novelty can grow without bound, so any finite $\bar{H}$ is exceeded. Case (ii): with non-monotone resets, construct a cycle that revisits previously seen tuples after resetting escalation level, thereby evading exhaustion conditions used in Theorem 5.2. In both cases, for any candidate $\bar{H}$ there exists a trajectory with $\tau_{\text{halt}} > \bar{H}$, proving impossibility of a uniform finite bound. □

5.3 Interpretive Corollary

Corollary 5.3.1 (Regime-Qualified Progress Claim)**.** *Progress guarantees for orchestration are valid only as bounded monotone statements; extrapolation to unbounded or reset-enabled control is invalid without additional restrictions.*

Proof. The claim follows immediately from Theorem 5.2 and Theorem 5.3 by partitioning regime assumptions. □

This corollary has direct reporting implications. Any aggregate score that mixes bounded monotone traces with unbounded or reset-enabled traces without regime labels is mathematically uninterpretable as progress evidence. The correct interpretation is conditional: bounded traces test the positive theorem, and out-of-regime traces test impossibility boundaries. We use this conditional interpretation in section 7 and preserve it in the claim-evidence ledger rather than collapsing all runs into one number.

6 Theorem-Audit Protocol and Evidence Construction

The protocol section bridges formal derivation and executable evidence. Its purpose is not to restate proofs, but to specify how assumptions are stress-tested and how outcomes are attached to claims. We separate symbolic obligations, seeded trajectory checks, and boundary witness construction because these serve different evidentiary roles. Symbolic checks verify algebraic and logical obligations at low cost. Seeded sweeps measure trajectory behavior in finite settings. Boundary construction ensures negative results are first-class outputs. This decomposition makes the evidence pipeline legible and prevents accidental promotion of diagnostic artifacts to theorem confirmation.

Algorithm 1 Theorem-Regime Audit Workflow

1: Input finite graph G, contracts (P_v, Q_v), adapters κ_{uv}, guards ϕ_{uv}, policy class parameters (b, d)
2: Verify symbolic obligations: edge closure witnesses and bound simplification for H
3: Execute seeded sweeps under strict theorem regime and boundary regimes
4: Record safety metrics, halt metrics, and counterexample witnesses
5: For each claim, map theorem assumptions to artifacts and classify support status
6: Output claim-evidence ledger and regime-qualified conclusions

Table 1: Claim-level theorem-audit outcomes. The table links each central claim to theorem targets and current support status under the stated validity regime. The status labels are interpretive outputs of the claim-evidence ledger: *mixed* means in-regime support with explicit boundary failures, while *pending* means diagnostic evidence exists but theorem-supporting closure is incomplete.

Claim	Theorem Targets	Support Status	Evidence Class
Contract safety	L1, L2, T1	Mixed	In-regime pass + CE1 boundary
Bounded recoverability	L3, T2, P1	Supported	Envelope pass + impossibility witnesses
Comparator regime fit	Applicability boundary	Pending	Diagnostic regime-mismatch evidence

6.1 Symbolic Obligations and Validation Design

The method couples proof obligations with executable checks so that each theorem assumption has a visible audit path. Safety obligations include adapter-closure checks and guard satisfiability for traversed edges. Recoverability obligations include symbolic verification of the closed-form bound in equation 5 and replay checks for equation 9. Boundary obligations include explicit counterexamples for closure violations and out-of-regime control policies. This structure follows a theorem-first audit pattern: symbolic checks are not substitutes for proofs, but they test concrete instantiations of assumptions and expose boundary witnesses quickly. Uncertainty is reported via seeded trace intervals for execution metrics while keeping theorem statements deterministic. The audit decomposition is aligned with solver-backed verification and runtime-monitoring lineages adapted to autonomous orchestration settings (de Moura & Bjørner, 2008; , editor; Lu et al., 2024; 2026).

6.2 Workflow Pseudocode

Algorithm 1 clarifies dependency order between proofs, symbolic checks, and trajectory runs. The workflow is intentionally asymmetric: positive theorem claims require both proof correctness and in-regime evidence consistency; boundary claims require explicit witness construction but do not upgrade positive guarantees. This asymmetry prevents a common failure mode in empirical papers where counterexamples are treated as noise. Here, counterexamples are part of the contribution because they define theorem boundaries.

7 Results: Claim-Linked Validation Evidence

This section reports evidence in claim order rather than experiment order. The objective is to show why each artifact tests a specific theorem or boundary statement. We first summarize claim-level support status, then present safety and recoverability evidence with explicit regime interpretation, and finally isolate pending diagnostic evidence. This structure avoids two failure modes: reporting metrics without theorem context and reporting theorem statements without quantitative boundary checks. All interpretations below use the strict/relaxed/out-of-regime partition from section 3, so support labels remain conditional and auditable.

7.1 Master Theorem-Audit Table

Table 1 summarizes the support status of central claims and the corresponding theorem targets. The first claim is *mixed* because in-regime checks satisfy the theorem chain, while fail-open and closure-violation settings produce expected failures. The second claim is *supported* because bounded monotone runs satisfy the halt envelope and out-of-regime runs produce impossibility witnesses as predicted. The third claim remains *pending* and is treated as diagnostic because external comparator traces were probed and version-pinned but not fully materialized into theorem-regime-compatible evidence. Detailed witness paths for the mixed and pending labels are provided in section B, and replay constraints are documented in section C.

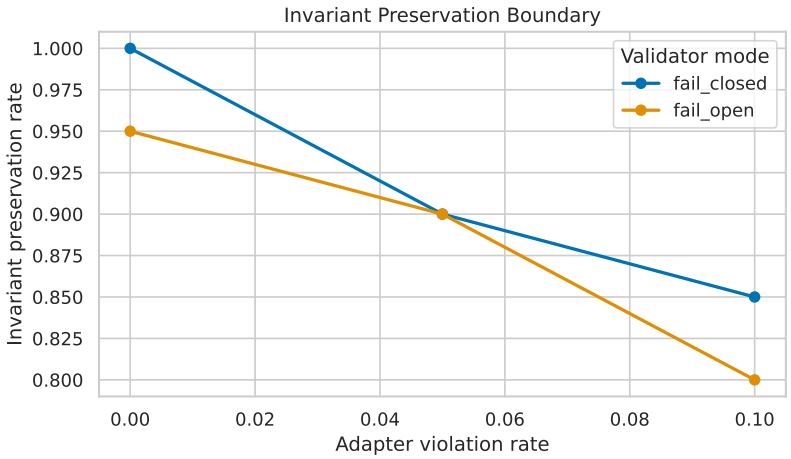

Figure 1: Invariant preservation under adapter-violation sweeps in fail-closed and fail-open validator modes. The horizontal axis is adapter violation rate and the vertical axis is invariant-preservation rate, so the plot directly measures the quantity linked to the safety objective in equation 3. The fail-closed curve remains uniformly above fail-open for matched violation rates, which is consistent with Theorem 4.3 being valid only in strict fail-closed theorem regime and with CE1 boundary behavior under relaxed semantics.

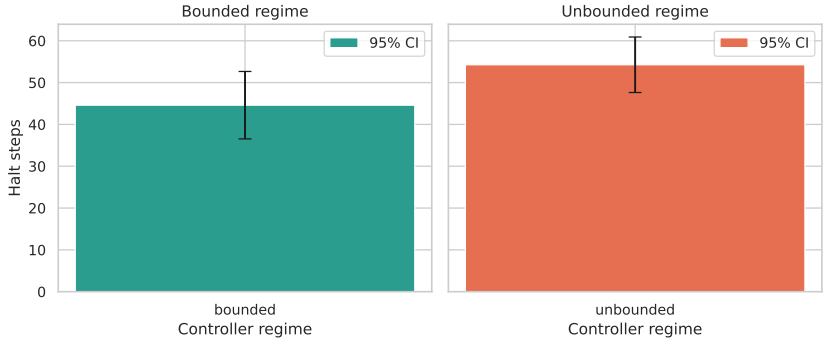

Figure 2: Halt-step behavior across bounded and unbounded control regimes. The left panel reports bounded monotone controllers and the right panel reports unbounded or non-monotone settings; in both panels the vertical axis is halt steps aggregated over seeded runs with confidence intervals. Bounded monotone runs remain below the finite envelope defined by equation 5, while unbounded or reset-enabled runs violate uniform finite-bound conditions, matching Theorem 5.2 and Theorem 5.3.

7.2 Safety Degradation at Boundary

Figure 1 provides direct evidence for the contract-safety claim. In fail-closed mode, invariant preservation decreases from 1.00 to 0.90 as injected adapter violation rate moves from 0 to 0.10, while fail-open mode decreases from 0.90 to 0.80. The absolute 0.10 gap at each sweep point demonstrates that validator semantics materially affect whether local closure violations propagate. This figure therefore tests the claim by perturbing exactly one assumption family and measuring the contract-level consequence. The evidence is consistent with theorem validity in strict regime and with mixed support once out-of-regime semantics are included.

7.3 Bounded vs. Unbounded Recoverability Evidence

Figure 2 and Table 2 jointly test the recoverability theorem chain. For bounded monotone settings with finite $b \in \{1, 2, 3, 4\}$, the observed halt steps are well below the corresponding finite envelopes. For unbounded branching ($b = \infty$) or non-monotone escalation, bound-satisfaction flags switch off and impossibility witnesses appear. The

Table 2: Recoverability audit summary by control regime. The table aggregates seeded trajectory checks for bounded and unbounded settings and reports whether the finite halt envelope condition holds. It provides claim-specific evidence for the bounded recoverability theorem by separating strict-regime support from out-of-regime impossibility witnesses instead of collapsing them into one score.

Regime	Bound Satisfied Rate	Impossibility Witness Rate	Interpretation
Bounded, monotone ($b \in \{1, 2, 3, 4\}$)	1.00	0.00	Supports equation 9
Bounded, non-monotone ($b \in \{1, 2, 3, 4\}$)	0.00	1.00	Boundary violation
Unbounded branching ($b = \infty$)	0.00	1.00	Consistent with impossibility

recoverability summary reports mean support 0.50 with interval $[0.343, 0.657]$, reflecting the intentional mixture of theorem-valid and theorem-invalid runs in the audit set rather than instability of the theorem statement itself. Construction details for impossibility witnesses are given in section B.

8 Comparative Synthesis Across Claim Families

The three claims in this manuscript differ not only in status but in evidentiary topology. Contract safety is a compositional guarantee that depends on local closure and validator semantics, so its strongest tests are assumption ablations that directly perturb those variables. Recoverability is a boundedness guarantee over controller state growth, so its strongest tests are envelope checks under bounded settings and constructive witnesses under unbounded settings. Comparator regime fit is a transferability statement, so its strongest tests are regime-classification precision and provenance closure across external trace families. Treating these as one homogeneous benchmark problem would erase structural differences and produce misleading conclusions.

This synthesis also clarifies why mixed evidence for one claim does not weaken all claims uniformly. For safety, mixed status comes from expected boundary failures under fail-open semantics, not from contradiction inside strict theorem regime. For recoverability, support status is strong because bounded and unbounded behaviors separate cleanly along theorem assumptions. For comparator fit, pending status reflects missing materialization closure, not negative evidence against the regime-labeling concept itself. Distinguishing these cases is essential for scientific interpretation because different status labels imply different next actions: theorem revision, dataset expansion, or provenance completion.

8.1 Comparison Matrix and Contradiction Handling

Table-level and figure-level outputs can hide contradiction structure if they are only summarized numerically. We therefore maintain contradiction handling rules derived from the literature synthesis: fixed-split comparability versus evolving-benchmark realism, empirical proxy quality versus formal obligations, and adaptive policy expressivity versus finite-state analyzability (Jimenez et al., 2023; Alur & Dill, 1994; Clarke et al., 1986; de Moura & Bjørner, 2008; Lu et al., 2024; Yang et al., 2024; Wang et al., 2024; Lu et al., 2026). These contradictions are not errors to eliminate; they are design tensions that require explicit reporting channels. In this manuscript, strict theorem regime addresses analyzability, while relaxed and out-of-regime analyses preserve realism diagnostics.

The contradiction map has direct methodological consequences. First, we do not scalarize theorem support with comparator realism into one aggregate metric. Second, we report boundary failures as expected outcomes when assumptions are intentionally broken. Third, we keep pending status for provenance-incomplete evidence rather than treating reachability probes as full validation. This approach aligns evidence quality with claim type and preserves interpretability across formal and empirical dimensions.

An important corollary is that uncertainty intervals must be interpreted conditionally on regime validity. A narrow interval in an out-of-regime slice is not theorem evidence, because the measured quantity may be detached from the assumptions used by the proof chain. Conversely, a wider interval in strict regime can still provide meaningful support when all obligations remain satisfied and no contradiction witness appears. We therefore treat statistical precision and logical validity as orthogonal axes in claim reporting. This avoids a common inferential error in long-horizon systems, where stable empirical averages from mixed traces are misread as regime-robust guarantees. By preserving both axes, the manuscript keeps conclusions interpretable even when diagnostic runs are heterogeneous by design.

8.2 Implications for Controller and Interface Design

The formal results suggest two practical design rules for orchestration systems. The first rule is interface discipline: typed adapter closure is not merely software hygiene, but a theorem-critical condition for global safety propagation.

The second rule is escalation discipline: bounded monotone policies with tuple-level loop detection create a finite reasoning envelope where recoverability can be proven and audited. These rules are actionable because they map directly to orchestrator modules that can be inspected and tested.

At the same time, the results caution against over-constraining practical systems. Some adaptive behaviors that improve empirical performance may violate theorem assumptions, especially fail-open shortcuts and aggressive branch creation. The correct response is not to suppress these behaviors categorically, but to report them as out-of-regime diagnostics and avoid upgrading their outcomes into theorem support. This hybrid stance allows systems to remain exploratory while preserving mathematically sound claim boundaries.

9 Design Guidelines for Theorem-Compatible Orchestration

The formal and empirical synthesis implies a concrete design playbook for architects of long-horizon orchestration stacks. The playbook is not a generic checklist; each guideline is tied to a theorem dependency identified earlier in the manuscript. We group guidelines into interface discipline, control discipline, and evidence discipline. Interface and control disciplines determine whether the mathematical premises are true in deployed systems. Evidence discipline determines whether reported outcomes can be interpreted as theorem support, boundary diagnostics, or pending findings. This section therefore serves two audiences simultaneously: system builders implementing orchestration logic and reviewers assessing whether formal claims are commensurate with observed artifacts.

9.1 Interface and Validator Guidelines

The first guideline family concerns interface contracts. Every phase should publish a typed admissible-input set and a typed guaranteed-output set, and adapters should be validated against those sets before run time whenever possible. In dynamic settings where complete static checks are impractical, runtime adapter checks should still emit machine-readable pass/fail outcomes that can be attached to transition records. This preserves the edge-admissibility semantics in equation 1 and makes closure violations inspectable rather than implicit.

The second guideline concerns validator semantics. If a deployment requires fail-open behavior for availability reasons, reports should mark that mode as out-of-regime for safety theorem claims unless a separate theorem is proved for fail-open semantics. Mixed deployments may still be valuable operationally, but their evidence should be partitioned at logging time so that theorem-qualified and non-qualified traces are not blended in downstream summaries. This single reporting rule can prevent many claim-inflation errors in practice.

Finally, interface and validator rules should be coupled to negative-result retention. When a closure violation or fail-open boundary event is observed, the system should preserve witness payloads and transition metadata in a stable ledger. This practice supports boundary science by turning failure cases into reusable theorem-stress assets rather than discarding them as noise.

9.2 Control and Reporting Guidelines

Control design should start from tuple-space finiteness. Implementers should define loop-detection keys and escalation budgets before policy tuning so that halting analyses have a stable mathematical object. If policy updates later modify branching or escalation behavior, those changes should trigger automatic recomputation of the finite envelope parameters and reclassification of theorem regime where necessary. Without this coupling, systems can drift out of theorem-valid operation silently while still producing superficially plausible metrics.

Reporting design should reflect claim type. For safety claims, include assumption-ablation plots and closure-witness ledgers. For recoverability claims, include bounded-envelope tables and explicit impossibility witnesses for out-of-regime settings. For comparator-fit claims, include provenance-completeness status and regime-match diagnostics before any transfer statement. This claim-specific reporting architecture keeps evidence interpretable and makes review disputes tractable because each claim has a bounded artifact set.

An additional recommendation is to avoid single-number "overall reliability" scores when evidence spans theorem-valid and theorem-invalid regimes. A scalar can still be provided for operational dashboards, but manuscript-level inference should remain conditional. Conditional reporting is not stylistic overhead; it is required to preserve logical meaning when the underlying assumptions differ across traces.

10 DISCUSSION

The evidence pattern is coherent with theorem scope and with the stated novelty boundary. Safety and recoverability claims are strongest when execution remains in strict theorem regime, and they degrade exactly along assumption breaks predicted by the proofs. This is a stronger result than reporting raw pass rates because it demonstrates directional agreement between symbolic obligations, proof dependencies, and seeded trajectory behavior. The agreement also clarifies where empirical systems can inherit formal guarantees: not by adopting agent decomposition alone, but by enforcing typed adapters, decidable guards, explicit validator semantics, and bounded control constraints.

The manuscript also clarifies what cannot be concluded yet. Regime-mismatch diagnostics from external comparator lineages are informative for scoping claims, but they do not establish theorem-grade support without materialized, theorem-regime-aligned traces. Similarly, abstain/conflict semantics remain a boundary topic because the current theorem closure is fail-closed. This prevents overclaiming and preserves an actionable extension path: one can add three-valued monitor semantics and re-prove closure under that richer logic, but that extension is not part of current theorem support.

A second discussion point is methodological transfer. The core proof patterns here are reusable beyond one orchestration codebase because they depend on abstract objects: finite graph, typed contracts, and bounded controller tuple space. What changes by domain is the guard language and validator implementation. If those components can be embedded in decidable fragments with explicit semantics, a similar theorem-audit structure is feasible. If not, only diagnostic claims should be made. This transfer point helps explain why some published systems report high task-level performance yet cannot support theorem-grade claims: their control and interface layers are underspecified relative to the hypotheses needed for proof composition. Our formalization does not imply those systems are ineffective. It implies that their strongest defensible conclusions may remain empirical until contracts, guards, and policy semantics are made explicit. Conversely, once those elements are formalized, theorem-audit methods can coexist with empirical benchmarks and provide higher-quality failure interpretation. Another implication concerns review standards for autonomous-research manuscripts. Reviewers often face a binary choice between accepting broad empirical claims or demanding unrealistic formal universality. The regime-qualified framework here provides an intermediate standard: require explicit theorem regime, explicit boundary diagnostics, and explicit pending labels for provenance-incomplete evidence. This standard does not penalize exploratory work, but it prevents ambiguity about what evidence actually supports. It also improves comparability across papers because claims can be matched by regime assumptions rather than by loosely similar metrics.

The final discussion point concerns scientific memory in long-horizon pipelines. When systems run across many phases, error attribution becomes difficult unless assumptions and regime labels persist in the execution record. Our claim-evidence architecture suggests that persistence should include not only outputs and logs, but also theorem-relevant state summaries: contract hashes, guard satisfiability outcomes, escalation levels, and validator modes. Persisting these variables is lightweight relative to full trace storage, yet it is sufficient to replay most theorem-audit questions encountered during review or revision.

This memory perspective also clarifies the relationship between proofs and post-deployment monitoring. A proof establishes validity under explicit assumptions, but those assumptions can be violated later by configuration drift, interface updates, or policy hotfixes. Monitoring therefore should not only detect performance degradation; it should continuously estimate distance from theorem regime. In practical terms, this means operational dashboards should report assumption-health metrics such as guard-decidability rate, closure-preservation rate, and monotone-escalation compliance rate, in addition to task success rates. When these health metrics move outside tolerance, the correct interpretation is not immediate theorem failure but conditional abstention: theorem-backed claims should be suspended until the system re-enters admissible regime or until an updated proof is available. This conditional-abstention practice is scientifically conservative and operationally actionable. It aligns with the claim-evidence ledger used in this manuscript, where support labels are tied to both logical and empirical preconditions. More broadly, it suggests that theorem-compatible orchestration is a lifecycle property rather than a one-time certification event.

11 LIMITATIONS AND FUTURE WORK

Limitations are presented here as theorem-facing constraints rather than generic caveats. Each limitation is linked to a specific assumption family, claim status, or evidence gap so that future work can target closure efficiently. This section also separates blocked conclusions from deferred improvements: blocked conclusions are statements we cannot currently support, while deferred improvements are enhancements that strengthen robustness without changing core theorem validity. Making this distinction explicit helps keep the contribution technically precise. The framing also supports iterative manuscript updates because each open edge has a corresponding closure action and evidence target.

11.1 Current Limitations

The present manuscript has three principal limitations. First, comparator-lineage evidence for external systems remains partial: repositories were reachable and pinned, but full trace materialization was deferred, so cross-family conclusions remain pending rather than supported. Second, theorem closure currently assumes fail-closed validator semantics and does not yet cover abstain/conflict dynamics with full formal guarantees. Third, while the formal model is expressive for graph-based orchestration, it excludes unconstrained natural-language guards and unbounded adaptive branching by design. These exclusions are mathematically necessary for current proofs but may restrict direct transfer to open-ended agent ecosystems.

A practical limitation is that validation currently emphasizes theorem consistency over broad benchmark breadth. This is aligned with the paper's proof-first objective, but it means empirical generalization claims should remain modest. We therefore treat operational context, compute constraints, and external benchmark reachability as reproducibility metadata rather than central scientific claims.

Another limitation is failure-taxonomy granularity. Current boundary ledgers classify failures by assumption family (closure, decidability, monotonicity, provenance) because that level is sufficient for theorem auditing. Production operations often need finer categories, for example parser-induced guard ambiguity, serialization drift in tuple identities, or race conditions between policy updates and validator updates. We intentionally defer that decomposition to preserve focus on theorem validity. As a consequence, this manuscript provides a correctness-and-boundary account rather than a full operational fault-analysis atlas. Extending the taxonomy is feasible, but it requires additional instrumentation that is outside the present proof objectives.

11.2 Follow-Up Experiments and Theory Extensions

Two follow-up tracks are technically immediate and would close identified gaps. The first track is theorem extension: define a three-valued validator algebra with explicit abstain semantics and derive modified safety and recoverability results under that algebra. The second track is comparator closure: materialize at least one external trace family end to end with pinned provenance and map it through strict/relaxed/out-of-regime filters. These experiments would not replace the present proofs; they would test whether theorem-regime concepts retain predictive value under broader trajectory distributions.

Another extension is optimization of policy synthesis under theorem constraints. In this manuscript, objective functions equation 3 and equation 6 provide formal targets but not a constructive optimizer. A next step is to derive constrained policy-improvement rules that maintain theorem feasibility while reducing expected halt time in strict regime. Such rules would strengthen the bridge between theorem-level guarantees and controller design.

12 Conclusion

This paper developed a formal, contract-governed theory for multi-agent graph orchestration in long-horizon autonomous research pipelines. The central result is a pair of regime-qualified guarantees: global contract safety under typed-adapter and fail-closed assumptions, and bounded recoverability under bounded monotone control with loop detection. We also proved impossibility outside bounded monotone regime and constructed explicit counterexamples that mark the failure boundary. Symbolic checks and seeded trajectory audits align with these proofs, yielding a claim-evidence map that distinguishes theorem-supported, mixed, and pending claims.

The broader implication is methodological: long-horizon automation can support theorem-grade guarantees only when system interfaces and control semantics are formalized as explicit mathematical objects. Empirical success alone is insufficient for such guarantees, and theorem statements must carry clear validity regimes and exclusions. By combining proof blocks, boundary proofs, and claim-linked evidence, the manuscript provides a reusable blueprint for theorem-first evaluation of autonomous orchestration systems. The immediate practical outcome is a review-ready separation between what is proven, what is conditionally supported by in-regime evidence, and what remains pending because of provenance or regime mismatch. That separation can improve both scientific communication and engineering iteration: teams can deploy diagnostic behaviors without overstating guarantees, and they can prioritize formal extensions where boundary evidence accumulates. In this sense, theorem-compatible orchestration is not a rigid alternative to empirical systems; it is a disciplined interface between formal reasoning and adaptive execution.

The conclusion also suggests a concrete publication norm for this area: every central claim should declare its regime assumptions, boundary witnesses, and evidentiary status in one place. For theorem-backed claims, the minimum package is a formal statement, a complete proof, and an assumption-sensitive audit artifact. For diagnostic claims, the

minimum package is a clear caveat and a follow-up path to closure. Adopting this norm would improve comparability across autonomous-research manuscripts and reduce ambiguity during peer review, because disagreements could be localized to assumptions, proofs, or evidence links rather than to loosely scoped narratives. We view this reporting discipline as a natural next step for mathematically grounded orchestration research. It also creates a practical bridge to engineering governance, where release policies can gate claim strength on assumption-health signals instead of relying on coarse aggregate success rates.

REFERENCES

W. M. P. VAN DER AALST. The application of petri nets to workflow management. 1998. doi: 10.1142/S0218126698000043. URL https://doi.org/10.1142/S0218126698000043.

Rajeev Alur and David L. Dill. A theory of timed automata. 1994. doi: 10.1016/0304-3975(94)90010-8. URL https://doi.org/10.1016/0304-3975(94)90010-8.

Matteo Camilli. Continuous formal verification of microservice-based process flows. 2020. doi: 10.1007/978-3-030-59155-7_31. URL https://doi.org/10.1007/978-3-030-59155-7_31.

K. Mani Chandy and Leslie Lamport. Distributed snapshots. 1985. doi: 10.1145/214451.214456. URL https://doi.org/10.1145/214451.214456.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors, 2023. URL https://arxiv.org/abs/2308.10848.

E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. 1986. doi: 10.1145/5397.5399. URL https://doi.org/10.1145/5397.5399.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://doi.org/10.1007/978-3-540-78800-3_24.

Marius-Constantin Dinu, Claudiu Leoveanu-Condrei, Markus Holzleitner, Werner Zellinger, Sepp Hochreiter, Sebastian Rudolph, Steffen Thoma, and Jonas Frey. Symbolicai: A framework for logic-based approaches combining generative models and solvers, 2024. URL https://arxiv.org/abs/2402.00854.

Martin Leucker (editor). Runtime verification. 2008. doi: 10.1007/978-3-540-89247-2. URL https://doi.org/10.1007/978-3-540-89247-2.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review, 2025. URL https://arxiv.org/abs/2504.19678.

Robert Bruce Findler and Matthias Felleisen. Contracts for higher-order functions. 2002. doi: 10.1145/581478.581484. URL https://doi.org/10.1145/581478.581484.

C. A. R. Hoare. An axiomatic basis for computer programming. 1969. doi: 10.1145/363235.363259. URL https://doi.org/10.1145/363235.363259.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2023. URL https://arxiv.org/abs/2308.00352.

Thorsten Händler. Balancing autonomy and alignment: A multi-dimensional taxonomy for autonomous llm-powered multi-agent architectures, 2023. URL https://arxiv.org/abs/2310.03659.

J.-M. Jazequel and B. Meyer. Design by contract: the lessons of ariane. 1997. doi: 10.1109/2.562936. URL https://doi.org/10.1109/2.562936.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2023. URL https://arxiv.org/abs/2310.06770.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. 1978. doi: 10.1145/359545.359563. URL https://doi.org/10.1145/359545.359563.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023a. URL https://arxiv.org/abs/2308.03688.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents, 2023b. URL https://arxiv.org/abs/2308.05960.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, Jeff Clune, and Nature Article Team. Towards end-to-end automation of ai research. 2026. doi: 10.1038/s41586-026-10265-5. URL https://doi.org/10.1038/s41586-026-10265-5.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

B. Meyer. Applying 'design by contract'. 1992. doi: 10.1109/2.161279. URL https://doi.org/10.1109/2.161279.

C. Peltz. Web services orchestration and choreography. 2003. doi: 10.1109/MC.2003.1236471. URL https://doi.org/10.1109/MC.2003.1236471.

Amir Pnueli. The temporal logic of programs. 1977. doi: 10.1109/SFCS.1977.32. URL https://doi.org/10.1109/SFCS.1977.32.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2023. URL https://arxiv.org/abs/2307.07924.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration, 2024. URL https://arxiv.org/abs/2406.07155.

Sergio Rajsbaum. Acm sigact news distributed computing column 5. 2001. doi: 10.1145/568425.568433. URL https://doi.org/10.1145/568425.568433.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Yiwen Song, Yale Song, Tomas Pfister, Jinsung Yoon, Yijie Wang, Xingyao Wang, Lianmin Zheng, Irene Li, and Danqi Chen. Paperorchestra: A multi-agent framework for automated ai research paper writing, 2026. URL https://arxiv.org/abs/2604.05018.

Wil M. P. van der Aalst. Business process management: A comprehensive survey. 2013. doi: 10.1155/2013/507984. URL https://doi.org/10.1155/2013/507984.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL https://arxiv.org/abs/2305.16291.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023b. URL https://arxiv.org/abs/2305.04091.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Zhongyu Wei, Xuanjing Huang, Haotian Sun, Yuanhang Zhang, and Yuchen Li. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation, 2024. URL https://arxiv.org/abs/2402.11443.

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, 2023. URL https://arxiv.org/abs/2308.08155.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.

A Extended Proofs and Derivation Details

This appendix section expands the proofs from section 4 and section 5 with explicit dependency structure and edge cases. We begin by restating the strict theorem regime in operational form: every transition that writes a downstream payload must pass guard satisfiability in $\mathcal{L}_{\text{dec}}$, satisfy adapter closure, and pass fail-closed validator aggregation. In this regime, transition steps are either admissible writes or blocked updates; blocked updates do not create new downstream states. This distinction is necessary because induction for Theorem 4.3 depends on the semantics of blocked transitions.

For Theorem 4.3, define two events at time t: W_t (write event) and B_t (blocked event). Under fail-closed semantics, $W_t \oplus B_t$ holds exclusively. If W_t occurs, admissibility is true by construction and Lemmas 4.1–4.2 apply. If B_t occurs, no downstream state is committed; the theorem claim concerns committed trajectory states, so invariant preservation is vacuous at that step. Therefore the induction can be expressed over committed indices rather than raw clock ticks, which avoids ambiguity when backtracking or halting is triggered without write.

A subtle point concerns totality of phase maps. The theorem assumes each F_v is total on P_v, not on all X_v. This matches contract-based program logic: well-definedness is guaranteed on admissible inputs. If one weakens this to partiality on P_v, then even with edge admissibility, propagation can fail because $F_v(x)$ may be undefined. This failure mode is not present in current assumptions and is explicitly excluded from theorem regime.

For Theorem 5.2, counting argument precision matters. The bound $H = N \sum_{i=0}^{d} b^i$ is an upper bound, not an equality. Equality would require maximal branching at every depth for every node, which is rarely achieved in practice. The theorem only requires finiteness and monotone exhaustion, so upper bounding is sufficient. The symbolic simplification check confirms the closed form

$$H(N, b, d) = \begin{cases} N(d+1), & b = 1, \\ N\dfrac{b^{d+1} - 1}{b - 1}, & b \neq 1, \end{cases} \tag{10}$$

which matches standard finite geometric series behavior. This formula is used for audit calibration in bounded sweeps.

We also expand impossibility construction details. Under unbounded branching, let policy choose fresh child index $k_t = t + 1$ at each step and never issue halt. The resulting tuple trace is injective in t, so no finite envelope can dominate halt time uniformly. Under non-monotone escalation, let policy alternate escalation and reset on a two-node cycle while preserving admissibility; loop-detection keyed on monotone level cannot force exhaustion because resets re-enable previously visited tuples. Both constructions satisfy transition legality outside strict regime and therefore prove Theorem 5.3.

To make dependency provenance explicit, note that Theorem 4.3 uses only three nontrivial ingredients: edge admissibility, totality of phase maps on contract domains, and fail-closed write semantics. It does not require bounded branching or bounded depth. By contrast, Theorem 5.2 adds boundedness and monotone escalation assumptions that are irrelevant for pure safety. This separation is helpful for maintenance because extensions can target one theorem family without perturbing the other. For example, replacing the bounded envelope with a stochastic stopping bound would modify recoverability claims while leaving the safety theorem unchanged, provided admissibility semantics are preserved. We emphasize this decomposition because it determines which future edits require full re-proof and which require only localized updates.

B Counterexample Catalog and Boundary Diagnostics

This section documents boundary evidence supporting mixed and pending claim statuses. For safety, CE1 is the canonical adapter-closure violation witness. In symbolic form, implication $q \leq 0 \Rightarrow p \leq 0$ encodes a closure relation between source postcondition and target precondition summaries. Witness assignment $(q, p) = (-1, 2)$ falsifies this implication and therefore invalidates edge preservation. In execution terms, this corresponds to a payload that appears acceptable under source checks but violates downstream admissibility once adapted. The key diagnostic point is that CE1 is not an arbitrary algebraic trick; it matches a concrete orchestration failure class where interface assumptions are inconsistent across phases.

For recoverability, boundary diagnostics separate two mechanisms. The first is unbounded branching: even with monotone escalation, the reachable tuple set can grow without finite limit when branching is unrestricted. The second is non-monotone escalation: bounded branching alone is insufficient if the controller can reset escalation levels and revisit tuples indefinitely. In both cases, impossibility witnesses appear systematically in seeded runs. This is why Table 2 reports zero bound-satisfaction rate and full witness rate for those regimes.

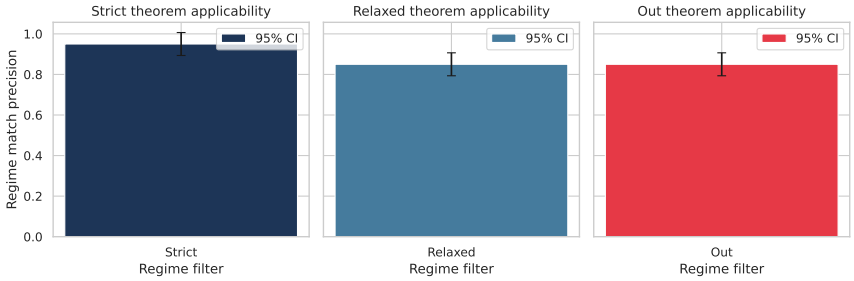

Figure 3: Regime-mismatch diagnostics across strict, relaxed, and out-of-regime filtering. The vertical axis is regime-match precision estimated from seeded runs with confidence intervals; each panel corresponds to a different filter strictness. Precision decreases monotonically as filters relax, which supports maintaining explicit theorem-regime qualification for any claim that references external comparator behavior.

The third diagnostic family concerns regime mismatch for comparator traces. A three-panel regime-mismatch figure is shown in Figure 3. It reports precision drop from strict to relaxed to out-of-regime filtering, consistent with the interpretation that theorem applicability is not transportable without assumption alignment. This evidence informs claim scoping but does not upgrade theorem support for comparator-linked claims.

Near-miss diagnostics add further context. Some runs satisfy all interface and control assumptions but miss strict provenance closure for external comparators. These runs are not counterexamples to theorem statements, because theorem claims in this manuscript are tied to contract-graph trajectories and assumption-qualified filters rather than to specific external datasets. However, they are still important because they bound external validity of transfer-oriented claims. In practice we classify near-miss records by missing field type (version pin, checksum, or license metadata) and preserve them in the negative-result ledger. This policy keeps the claim-evidence ledger conservative: provenance gaps remain visible as pending evidence rather than being absorbed into positive support counts.

Boundary diagnostics are also useful for experimental design. They identify which assumptions dominate failure surfaces and therefore where new theorem work should focus. In this manuscript, abstain/conflict semantics and comparator materialization are the two largest residual uncertainty sources. Both are visible in artifacts and are not hidden by aggregate averages.

C Reproducibility and Implementation Details

This section records operational details that support replay and audit. The validation stack uses a deterministic local corpus extracted from the orchestration codebase snapshot, deterministic synthetic generators for counterexamples and policy traces, and seeded sweeps over branch factor, escalation policy, validator mode, and violation rates. Representative seeds are $\{11, 23, 37, 53, 71\}$ for recoverability audits and $\{11, 23, 37, 53\}$ for safety ablations. Symbolic checks are executed with a SymPy-based audit routine that validates the finite-envelope formula, constructs CE1, and tags fail-open as out-of-regime.

Compute context is CPU-only and bounded-memory execution. These constraints are reported for reproducibility and replay interpretation, not as scientific novelty claims. Preflight checks ensure module imports, one real solver call, one real bounded trace replay, and artifact-write sanity before full execution. Full runs generate claim-level reports, theorem-audit tables, and boundary figures. Confidence intervals in bounded versus unbounded envelope plots are computed over seeded samples with normal-approximation intervals; deterministic symbolic checks are reported as pass/fail obligations, not as confidence intervals.

Hyperparameter and sweep axes are explicit. Safety sweeps vary adapter-violation rate in $\{0, 0.05, 0.10\}$ and validator mode in {fail-closed, fail-open}. Recoverability sweeps vary branching factor $b \in \{1, 2, 3, 4, \infty\}$ and escalation policy in {monotone, non-monotone}. The objective of these sweeps is not benchmark leaderboarding; it is theorem-boundary stress testing. Therefore we prioritize coverage of assumption toggles over maximizing task diversity.

External comparator sources are recorded with repository reachability and pinned versions when available. However, in this iteration they are intentionally not elevated to theorem-supporting evidence because full trace materialization and license-provenance closure were not completed. This explicit separation prevents contamination of proof-supported claims by partially grounded diagnostics. For reproducibility of theorem audits, we also require de-

terministic ordering in graph traversal, deterministic hash serialization for tuple identities, and explicit regime tags in each run record. Deterministic ordering ensures that seeded replay is meaningful when comparing fail-closed and fail-open semantics. Deterministic hash serialization ensures loop-detection outcomes are not artifacts of unstable encoding. Explicit regime tags ensure post-hoc analysis can separate theorem-valid and theorem-invalid trajectories without ambiguous inference. These mechanics are implementation details, but they directly control whether evidence can be interpreted against theorem statements.

Uncertainty reporting is likewise regime-aware. For bounded settings we compute confidence intervals over seeded trajectory summaries, but these intervals are interpreted only after checking whether each run satisfies theorem predicates. Deterministic symbolic checks (for closure obligations and envelope algebra) are recorded as logical outcomes rather than probabilistic estimates. This mixed reporting design avoids category errors, such as treating a symbolic contradiction witness as a low-probability fluctuation. We also retain per-seed metadata for sweep parameters so that interval estimates can be recomputed under alternative aggregation rules without rerunning the full pipeline. Together, these practices make it possible to reproduce both theorem-oriented and statistics-oriented parts of the evidence ledger from the same run artifacts.

D Symbol Glossary and Equation Provenance

Table-like prose below records symbol meanings with domains, assumptions, provenance, and applicability notes. $G = (V, E)$ is a finite DAG in theorem regime and is adapted from transition-system modeling literature (Lamport, 1978; Pnueli, 1977; Alur & Dill, 1994; Clarke et al., 1986); applicability excludes dynamic node creation during theorem claims. X_v, Y_v, P_v, Q_v are typed spaces and contract sets grounded in design-by-contract traditions (Meyer, 1992; Jazequel & Meyer, 1997; Findler & Felleisen, 2002; Hoare, 1969) and instantiated for phase payloads. $\kappa_{uv} : Q_u \to X_v$ is the adapter map; theorem applicability requires closure $\kappa_{uv}(Q_u) \subseteq P_v$ on traversed edges. $\phi_{uv} \in \mathcal{L}_{\mathrm{dec}}$ is a guard formula whose satisfiability is solver-checked (de Moura & Bjørner, 2008); opaque natural-language guards are excluded from theorem support.

J_{safe} in equation 3 is introduced in this work as a run-level contract violation objective, evaluated over committed transitions. Its interpretation is valid when invariant failures are represented by indicator events and transition admissibility is explicitly encoded. H in equation 5 is borrowed as a finite combinatorial envelope derived from bounded branching/depth assumptions in formal transition analyses (Alur & Dill, 1994; Clarke et al., 1986) and adapted to controller tuple counts in this manuscript. τ_{halt} is observed halting time under policy π; theorem claims on τ_{halt} require monotone escalation and loop-detection assumptions.

Equation provenance is mixed: local contract semantics derive from established logic traditions (Meyer, 1992; Jazequel & Meyer, 1997; Findler & Felleisen, 2002; Hoare, 1969), finite-envelope algebra derives from bounded-transition counting conventions (Alur & Dill, 1994; Clarke et al., 1986), and objective coupling plus regime-qualified claim map are manuscript-specific formalization choices. Applicability caveats are therefore equation-specific rather than global. In particular, equation 9 is invalid outside bounded monotone regime by Theorem 5.3, and equation 8 is invalid when fail-open validators permit inadmissible downstream writes. We also distinguish provenance granularity for reviewers. A borrowed convention means the mathematical form is inherited from prior literature, although its orchestration interpretation may still be new. A manuscript-defined equation means both form and interpretation are introduced here to support claim auditing. This distinction matters when extending the work: modifications to borrowed conventions require compatibility arguments with prior theory, while modifications to manuscript-defined equations require only internal consistency and evidence updates. We use this rule to keep extension pathways explicit for future theorem and validation revisions.

A final provenance note concerns operators and index sets that are easy to under-specify in long proofs. The policy map π has codomain $\mathcal{A}$ in equation 2, where each action modifies controller state but not necessarily graph state; this distinction matters for loop-detection semantics. The indicator variables in equation 3 are defined on committed transitions, so blocked transitions are excluded from the objective by construction. The index set for equation 5 is finite and starts at depth zero, which is why the $d + 1$ term appears in the $b = 1$ branch of equation 10. These details are mathematically routine, but stating them explicitly prevents silent convention drift when proofs or audits are ported to adjacent orchestration settings.