SUCCESS IS NOT COMPLETION: GOAL-CLOSURE MONITORING AND CORROBORATED CONTRACTS FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

An API can report success even when an agent has not achieved the user's goal. Repeating a check does not resolve this ambiguity if the check shares the original failure cause. We study *post-action goal-closure identification*: when does the available evidence justify declaring a task complete? We introduce GOALSEAL-CG, a training-free monitor that represents ways the goal may still be open, selects observations that distinguish them from completion, and admits only recovery actions that are safe across the remaining alternatives. The resulting rule is sound relative to an explicit goal contract and fault model; indistinguishable omitted worlds remain an impossibility boundary. We introduce GOALCOMMITBENCH to test safety and useful completion jointly under tool, evidence, contract, and recovery faults. On an exact eight-model matched grid of 5,120 episodes (1,024 task–fault–model cells $\times$ five policies), GOALSEAL-CG v3 is the only policy that issues success declarations while combining zero observed dangerous declarations, 100% declaration precision, and 85.0% goal closure, the highest observed. This compares with 5.4% danger and 82.7% closure for VTC-style verification and 5.7% danger and 80.5% closure for RAC/Saga-style compensation. A disjoint prospective 45-task ToolBench-X retest likewise reduces faulty danger from 37.8% for the upstream agent to 0%, while raising clean safe closure from 57.8% under a strict fixed gate to 84.4% ($+26.7$ points) and lowering escalation from 41.5% to 14.1% (-27.4 points); all six pre-specified criteria pass across three model families. Matched mechanism and stateful cross-node studies attribute the safety gain to closure admission and show successful recovery or containment outside a deliberately indistinguishable common-mode control. QUORUMCONTRACT recovers required-clause recall to 100% at zero added danger, eliminating the 1.4-point danger increase of raw contracts and narrowing their closure deficit to the oracle by 9.8 points. Together, these results establish training-free post-action closure monitoring across model families, public workflows, and cross-node stateful services within explicit contracts and registered fault/evidence envelopes.

1 INTRODUCTION

For a tool-using agent, executing the right call is not sufficient to complete a task. A booking endpoint may acknowledge a replacement ticket that was never issued; a refund may be recorded without reaching the customer. These failures matter because agents act on persistent external state and report outcomes on which users rely. Completion is therefore a claim about the resulting world, not simply a property of the tool response.

Consider an agent changing a flight. After cancellation of the old ticket, the replacement endpoint returns success. Two states remain possible: the new ticket exists, or only the cancellation took effect. Blind retry can introduce a duplicate booking or charge; reading the same stale backend can repeat the misleading acknowledgement. The agent needs evidence that distinguishes these states before reporting "change complete." This example motivates our question: *when is goal closure identifiable from the available evidence, and what should the agent do when it is not?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

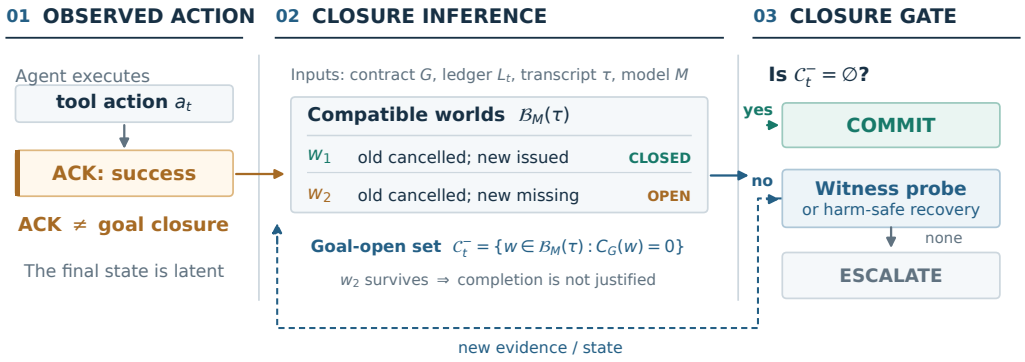


Figure 1: GoalSeal's post-action loop. An acknowledgement does not establish completion. Given contract G, ledger L_t, transcript τ, and model M, the monitor commits only when no compatible goal-open world remains. Otherwise it probes, safely recovers, or escalates. Dashed arrows return updated evidence or state for re-evaluation. Soundness is relative to the supplied contract and model.

Prior work studies silent tool errors (Sun et al., 2024), false-success declarations (Advani, 2026), missing tools and information (Treviño et al., 2025), reflective correction (Liu et al., 2026b), dynamic replanning (Zhu et al., 2026), learned recovery (Zhang et al., 2026), and failure attribution (Zhang et al., 2025). Three issues remain entangled. First, tool-level success is distinct from task-level semantic closure. Second, repeated verification does not provide independent evidence when channels share a failure cause. Third, recovery choices differ in cost, reversibility, and tail risk.

Our approach treats each compatible world in which the goal remains unmet as a counterexample to completion. GoalSeal-CG uses these counterexamples to choose informative checks and to rule out recovery actions that could cause harm in another compatible world. A task is admitted as complete only after all modeled counterexamples have been eliminated; otherwise the monitor gathers evidence, recovers, or escalates (Figure 1). The rule exposes two assumptions that verification alone cannot repair: the contract must express the user's goal, and the evidence model must include the realized failure. We test these assumptions separately through automatic contract construction and fault-bounded witness corroboration.

Our contributions are:

1. **A completion criterion and monitor.** We formalize goal-closure identification, characterize its observational limit and maximal sound admission rule, and implement the rule in GoalSeal-CG with counterexample-guided probing and harm-bounded recovery.
2. **A benchmark and controlled evaluation.** We introduce GoalCommitBench and evaluate safety jointly with autonomy across nine model configurations. On an exact eight-model intersection, GoalSeal-CG v3 is the only success-declaring policy with 0% danger, 100% declaration precision, and the highest observed closure (85.0%); VTC-style and RAC/Saga-style Agent adapters record 5.4%/5.7% danger and 82.7%/80.5% closure. On held-out ToolBench-X, v3 reduces upstream faulty danger from 37.8% to 0% and raises clean safe closure over a strict fixed gate from 57.8% to 84.4%.
3. **Robust corroboration.** QuorumContract raises required-clause recall from 70.3% to 100% with zero added danger. Across 2,592 witness episodes and four executable service/cross-node studies, GoalSeal eliminates registered non-common dangerous false success while preserving recovery or safe containment.

2 Related Work

Failure detection and recovery. Sun et al. (2024) expose success-shaped faulty outputs; FAIL-TaLMs and FeasiGen test missing tools or infeasible requests (Treviño et al., 2025; Cheng et al., 2026), while ReflecTool-Bench and AgentCheck evaluate reflective correction and replayed MCP

faults (Liu et al., 2026b; Mazumder & Lia, 2026). ReIn, structured reflection, Fission-GRPO, ToolMaze, and FLARE study test-time or learned repair/replanning (Kim et al., 2026; Su et al., 2026; Zhang et al., 2026; Zhu et al., 2026; Zhao et al., 2026). GOALSEAL-CG instead targets post-action closure ambiguity after the required tools were available and may have returned success.

False success is common and trace judges remain weak (Advani, 2026); learned trajectory/process evaluators offer another signal (Sun et al., 2026; Li et al., 2026; Wang et al., 2026). Telemetry-based repair and pre-execution policy gates cover complementary faults (Dubey, 2026; Reddy et al., 2026). We instead decide task-level admission under fresh but potentially common-cause evidence, reporting safety and abstention.

Contract and solver-based defenses govern a complementary decision boundary. STPA-derived tool-sequence and data-flow specifications constrain hazardous calls before execution, while SMT policy checking blocks planned calls that violate logical operational policies (Doshi et al., 2026; Winston et al., 2026). TraceGrant connects trusted intent, admitted evidence, concrete effects, and completion through a lifecycle contract (Liao et al., 2026). GoalSeal's distinct object is the post-action epistemic question: after a success-shaped result, do any contract-consistent open worlds remain, including worlds in which evidence sources share a fault? These approaches can supply clauses for G or transitions for M; possible-world closure determines when their post-action evidence is sufficient. Mind the GAP independently shows why safe text is not evidence of safe tool effects (Cartagena & Teixeira, 2026); our metric separates realized harm from the later decision to declare completion.

VIGIL separates hidden-state completion from terminal commitment and is an evaluation protocol rather than a recovery policy (Chen et al., 2026a). Goal-Autopilot requires executable gates before `DONE` (Deng, 2026); verify-gated completion uses read-only admission and recovery packets (Nguyen & Tran, 2026). Our distinct target is closure *identifiability* when checks can share stale or common-cause failures: compatible worlds determine distinguishing probes and safe recovery. Besides the decision-object comparison in Appendix B, we report a same-benchmark exploratory comparison with explicitly provenance- labeled ports of the two runnable decision rules.

Transactional agent execution. ToolGate and verified calls gate state transitions using contracts, postconditions, and idempotency (Liu et al., 2026a; Mansoor et al., 2026); RAC and DART address compensating or semantically admissible recovery (Perera et al., 2026; Yang et al., 2026); Atomix and Cordon provide semantic transaction machinery (Mohammadi et al., 2026; Chen et al., 2026b). Our orthogonal question is when accumulated, possibly correlated evidence makes a task-level user-visible success declaration admissible. We claim neither database isolation nor general rollback.

Risk control and calibration. Recent synthesis identifies semantic specification and verifier tax as coupled obstacles (Dantas et al., 2026); we measure these two surfaces separately. Selective prediction and conformal risk control trade coverage for error under explicit sampling assumptions (Geifman & El-Yaniv, 2019; Angelopoulos et al., 2024; Bai & Jin, 2026; Yu & Liu, 2026). Our object is a post-execution completion declaration with non-waivable structural obligations. Exact binomial intervals diagnose, but do not validate, the controlled sampling model (Clopper & Pearson, 1934); soundness remains conditional on the supplied contract and world model.

Stateful evaluation. τ-bench, τ^2-bench, and ToolBench-X evaluate stateful tool workflows and recoverable hazards (Yao et al., 2025; Barres et al., 2025; Tian et al., 2026); procedure-aware and Near-Miss evaluations expose invalid success paths (Cao et al., 2026; Rabinovich et al., 2026). GOAL-COMMITBENCH intervenes after success-shaped acknowledgements to test closure identification, correlated evidence failure, and safe recovery.

3 PROBLEM FORMULATION

3.1 GOAL CLOSURE AND POSSIBLE WORLDS

Let a task contract be $G = (P, Q^+, Q^-, I)$. P is a typed set of action preconditions; Q^+ and Q^- are respectively sets of required and forbidden Boolean predicates over the typed terminal state; and I is a set of Boolean predicates over the complete state–effect trace. Predicates carry their resource scope and argument types, so clauses over different bookings or accounts are not interchangeable. For

complete world w, including the true state, effect history, external changes, and tool-fault variables, closure is

$$C_G(w) = \bigwedge_{q \in Q^+} q(s_T(w)) \wedge \bigwedge_{q \in Q^-} \neg q(s_T(w)) \wedge \bigwedge_{i \in I} i(s_{0:T}(w)). \tag{1}$$

An episode is a *dangerous false success* when the agent calls `commit` while $C_G(w) = 0$.

Let M be an executable transition, fault, and observation model. The action–observation transcript τ induces the belief set

$$\mathcal{B}_M(\tau) = \{w \in \mathcal{W} : w \text{ is consistent with } M \text{ and } \tau\}. \tag{2}$$

Goal closure is *identifiable* at τ iff every member is closed. Each $w^- \in \mathcal{B}_M(\tau)$ with $C_G(w^-) = 0$ is a *closure counterexample*. This definition represents shared-cache and provider-wide faults directly in w; it does not gain confidence merely by naming two correlated endpoints.

Proposition 1 (observational barrier). Suppose a closed world w^+ and open world w^- induce the same transcript distribution under every allowed observation policy. Any monitor that commits with positive probability on that transcript has positive dangerous-false-success probability in at least one of the two worlds. Thus no prompting or confidence score can replace a distinguishing observation or an explicit escalation.

3.2 Actions and objective

At each step the agent chooses from observation, retry, tool switching, compensation, rollback, replanning, escalation, and commit. A probe partitions possible worlds by its outcome. An intervention transforms every possible world and may incur cost $c(a)$ and harm $h(a, w)$. Each modeled world has a registered positive weight $\mu(w)$; absent a justified fault prior, all worlds receive unit weight. These weights rank probes and repairs but never change the commit criterion. In the finite reference implementation, a probe's separated pair mass is $\sum_{w^+,w^-} \mu(w^+)\mu(w^-)\mathbf{1}[p(w^+) \neq p(w^-)]$, summed over currently compatible closed–open pairs. An intervention's closure-mass gain is $\sum_w \mu(w)[C_G(a(w)) - C_G(w)]$. The former is divided by registered probe cost, latency, and probe harm; the latter by intervention cost. Harm is a domain-registered nonnegative severity, not a probability: the selector admits a only when $\max_{w \in \mathcal{B}_M(\tau)} h(a, w)$ is within budget. The conformance grid uses a zero budget and binary harm (0 safe, 1 budget violation); deployments may replace it with a registered domain risk scale. Commit is admissible only when no closure counterexample remains.

4 GoalSeal-CG

4.1 Representing unresolved goals

The monitor starts from a practical question: what could still be wrong despite the observed acknowledgement? In the flight example, a missing replacement ticket and a duplicate charge require different evidence and different repairs. Counterexamples make these alternatives explicit. The runtime effect ledger records intended transitions, acknowledgements, resource identities, versions, idempotency keys, reversibility, compensation handles, and observed outcomes. An explicit-state enumerator or SMT backend combines this ledger with G and M to produce minimal open worlds consistent with τ. A counterexample includes its violated clauses, required common-cause faults, probes that distinguish it from closed worlds, and interventions that may repair it. An observation outside the modeled outcome support triggers escalation rather than silently deleting all worlds.

4.2 Choosing distinguishing evidence

For each closed–open pair $u = (w^+, w^-)$, probe p covers u when $p(w^+) \neq p(w^-)$. A non-adaptive witness protocol is a probe set covering every such pair:

$$P^* = \arg\min_{P' \subseteq P} \sum_{p \in P'} c(p) \quad \text{s.t.} \quad \forall (w^+, w^-),\ \exists p \in P' : p(w^+) \neq p(w^-). \tag{3}$$

This is weighted set cover over closed–open pairs and is NP-hard. We use exact enumeration as a small-instance oracle and the standard cost-normalized greedy cover as a scalable non-adaptive approximation.

Proposition 2 (greedy witness cost). Let U be the nonempty set of closed–open pairs and suppose a finite-cost witness cover exists. The implemented greedy rule, which maximizes newly separated pairs per unit positive cost, returns a witness of cost at most $H_{|U|}\,\mathrm{OPT} \le (1+\ln|U|)\,\mathrm{OPT}$, where OPT is the minimum non-adaptive witness cost.

The adaptive policy selects the next probe by separated pair mass per unit cost, conditions $\mathcal{B}_M(\tau)$ on the observed outcome, and repeats. A cheap success-shaped acknowledgement has zero value when it partitions no closed–open pair. Proposition 2 applies to the non-adaptive cover, not to this contingent decision tree.

An active witness binds its clause, resource, state version, request nonce, freshness epoch, authority, and evidence group. Shared-cause, stale, replayed, mismatched, revoked, or invalidly signed observations are rejected before conditioning the belief set; Appendix G gives an exhaustive development conformance test.

4.3 COUNTEREXAMPLE-GUIDED RECOVERY

GOALSEAL-CG projects candidate repairs across all remaining worlds, rejecting a retry if an already-committed world would incur duplicate or irreversible harm. It prefers an informative probe, then the harm-admissible repair with highest closure-mass gain per cost, and otherwise escalates. Each mutation invalidates state evidence and restarts counterexample generation. Appendix G.7 tests the verification/retry/switch/rollback/escalation branches, not their production frequencies.

If valid witnesses disagree, goal-completing mutations are forbidden; only a reversible containment action that weakly reduces registered risk in every compatible state and strictly reduces it in one is allowed. Evidence is then invalidated and re-probed; persistent disagreement escalates without a completion claim.

Algorithm 1 Counterexample-guided goal-closure monitoring

Require: Contract G; model M; transcript τ; ledger L; probes $\mathcal{P}$; recoveries $\mathcal{A}$; harm budget β
Ensure: COMMIT with a closure certificate, or ESCALATE with unresolved clauses

$\mathcal{B} \leftarrow \textsc{CompatibleWorlds}(G, M, L, \tau)$
loop
  if $\tau \notin \textsc{Support}(M)$ **then**
    return $\textsc{Escalate}\big(\textsc{Unresolved}(\mathcal{B}, G), L\big)$
  $\mathcal{B}^- \leftarrow \{w \in \mathcal{B} : C_G(w) = 0\}$
  if $\mathcal{B}^- = \varnothing$ **then**
    return $\textsc{Commit}\big(\textsc{Certificate}(G, M, L, \tau)\big)$
  $\mathcal{P}_\tau \leftarrow \{p \in \mathcal{P} : \textsc{Separates}(p, \mathcal{B})\}$
  if $\mathcal{P}_\tau \neq \varnothing$ **then**
    $p^* \leftarrow \arg\max_{p \in \mathcal{P}_\tau} \Delta_{\mathrm{pair}}(p; \mathcal{B})/b(p)$
    $o \leftarrow \textsc{Observe}(p^*)$
    if $\neg\textsc{Valid}(o, L, M)$ **then** ▷ freshness, scope, nonce, authority, fault domain
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \{p^*\}$; **continue**
    $\tau \leftarrow \tau \oplus (p^*, o)$; $\mathcal{B} \leftarrow \textsc{Condition}(\mathcal{B}, p^*, o)$
    continue
  $\mathcal{A}_\beta \leftarrow \{a \in \mathcal{A} : \max_{w \in \mathcal{B}} h(a, w) \le \beta \ \wedge\ \Delta_{\mathrm{close}}(a; \mathcal{B}) > 0\}$
  if $\mathcal{A}_\beta = \varnothing$ **then**
    return $\textsc{Escalate}\big(\textsc{Unresolved}(\mathcal{B}, G), L\big)$
  $a^* \leftarrow \arg\max_{a \in \mathcal{A}_\beta} \Delta_{\mathrm{close}}(a; \mathcal{B})/c(a)$
  $(\tau, L) \leftarrow \textsc{Execute}(a^*, \tau, L)$
  $\tau \leftarrow \textsc{InvalidateStateBoundEvidence}(\tau, a^*)$
  $\mathcal{B} \leftarrow \textsc{CompatibleWorlds}(G, M, L, \tau)$

Proposition 3 (model-relative soundness). If G is complete, M includes the realized transition, faults, and observations, and GOALSEAL-CG commits only when $\mathcal{B}_M(\tau) \subseteq \{w : C_G(w) = 1\}$, then dangerous false success is impossible relative to (G, M): the realized world remains in the belief set. Contract omissions, model misspecification, and opaque intra-call violations break this conditional guarantee.

Proposition 4 (maximal sound admission). For fixed (G, M), this closure rule is the unique pointwise-maximal sound rule measurable from the same transcript: admitting any additional reachable transcript commits in a compatible open world. Randomization cannot enlarge the admissible region (Appendix D).

Operational assurance envelope. Certificates expose the provenance/coverage of G and the fault/evidence envelope of M. Unattested premises receive a bounded or unverified label, never an unconditional safety claim (Appendix B).

4.4 IRREVERSIBLE-ACTION GATE AND RESIDUAL RISK

Before an irreversible action, the runtime requires all predecessor counterexamples to be eliminated and rejects any projected transition violating Q^- or I. A stable ledger key is injected when a non-idempotent tool supports it; otherwise the action escalates. For incompletely modeled environments, a secondary statistical layer measures selective risk on held-out causal templates and hidden fault compositions. It never treats parameterized trajectories from one workflow as independent evidence.

Selective closure admission. The v3 gate partitions unresolved diagnostics into *hard* and *soft* gaps. Missing required calls or fields, invocation failure, negative inner status, ungrounded final answers, and modeled correlated failures are hard and never waived. A soft auxiliary-field gap may be waived only if all required tools ran and a terminal producer without a hard gap grounds the final state. Writing $H(\tau)$ and $S(\tau)$ for the two gap sets, $A_{\text{all}}(\tau)$ for complete required-tool invocation, and $T^{\neg H}_{\text{term}}(\tau)$ for such a grounded terminal producer, the implemented admission rule is

$$a^{v3}_{\text{sel}}(\tau) = \mathbf{1}[H(\tau) = \varnothing]\,\mathbf{1}[S(\tau) = \varnothing \ \vee \ (A_{\text{all}}(\tau) \wedge T^{\neg H}_{\text{term}}(\tau))].$$

The gate waives redundancy, not structural failures. Its operating point is frozen on development data before held-out evaluation; a zero auxiliary threshold removes only a confidence cutoff, not the gate.

For providers returning terminal assistant text instead of a `commit` call, v3 converts a successful tool-free response into a *commit proposal* subject to the same gate; it cannot commit directly or mutate state. This is the sole policy change in the matched Agent extension (Table 1).

4.5 AUTOMATIC CONTRACTS

The oracle-contract track isolates monitoring and recovery. In the automatic track, models compile the instruction, initial state, and tool schemas into typed clauses and transition constraints. Against an executable oracle we measure clause precision/recall, counterexamples, forbidden effects, and invariant violations. Parse failures, invented keys, and non-executable contracts fail closed, separating search failure from clause omission.

Corroborated contract construction. QUORUMCONTRACT retains an exact typed clause if at least k fixed-registry model configurations emit it (one vote each). Invalid candidates abstain without shrinking the denominator; unsupported required classes fail closed. Among $k \in \{1, \ldots, 9\}$, development counterexample accuracy under zero danger selects $k = 2$ as hidden primary, also preserving zero danger in fixed-k leave-one-configuration-out checks. Union ($k = 1$) and strict majority ($k = 5$) remain secondary. This development choice is not hidden-generalization evidence.

Distinct from QUORUMCONTRACT's pre-execution clause aggregation, witness quorum corroborates post-action state evidence across auditors and is evaluated separately.

5 GOALCOMMITBENCH

What is evaluated. GOALCOMMITBENCH is an executable causal testbed for the gap between a successful tool acknowledgement and actual task completion; it is not a prevalence sample of production failures. Consider a travel agent asked to issue a replacement ticket before cancelling the original. The same success-shaped transcript can mean that both steps occurred or that only cancellation occurred; a ticket record distinguishes them, whereas blind retry may duplicate a charge. This closed/open pair, its distinguishing probes, and safe recoveries form one *counterfactual closure case*.

Case construction. The fixed pipeline is workflow → goal contract → transition/fault graph → ambiguous worlds → witnesses/recoveries. Each workflow specifies an initial state, three or four typed actions, terminal requirements, full-trace invariants, and an irreversible effect. The compiler expands action, observation, external-mutation, and compensation outcomes. A case is valid only if closed and open worlds share a weak-protocol transcript but a registered probe separates them; executable reachability and closure checks reject vacuous cases. Records retain the contract, state trace, causal fault variables, interventions, and probe costs.

The catalog contains 32 causal templates and 320 executable instances, split 8/8/16 templates across calibration, development, and hidden test; the full composition appears in Appendix 7.

Coverage and structural splits. The four domains are balanced. Faults span explicit failure, silent/partial/ wrong/duplicate effects, stale evidence, external revocation, and failed compensation. Hidden templates add unseen fan-out/join, split-settlement, identity, quorum, and multi-resource motifs; no template crosses a split. The main grid freezes one hidden task per template and crosses 16 templates, eight fault schedules, nine model configurations, and three policies (3,456 episodes; 1,152 matched triples). Fault locations are hash-bound before execution. Simulator state—not an LLM judge—determines closure, dangerous false success, harm, and invariant violation; coverage, escalation, and cost expose vacuous abstention. A blinded 24-item development audit obtains 23/24 exact contract agreement before adjudication. Appendix E gives the full schema, fault ontology, generation checks, and audit boundary.

6 EXPERIMENTAL DESIGN

We test whether monitoring prevents false completion without vacuous abstention, which mechanisms cause the change, and how contract or evidence failures alter the result. Appendix 6 maps each claim to its evidence.

Pairing and baselines. Comparisons pair model, task, world, fault location, and seed while holding the contract, permissions, recovery primitives, and budgets fixed. Seven frozen policies cover acknowledgement/retry, verification, reflection/replanning, Verified Tool Calls (Mansoor et al., 2026), RAC (Perera et al., 2026), DART (Yang et al., 2026), and Atomix/Cordon-style transactions (Mohammadi et al., 2026; Chen et al., 2026b). We label mechanism-equivalent adapters separately from author code. Certified/uncertified and same/independent-evidence contrasts isolate the gate, evidence access, and recovery; non-deployable oracles provide upper bounds.

Models, external test, and inference. Nine routes span six families (GPT, Claude, Gemini, GLM, DeepSeek, and Qwen). The prospective supportive test freezes v3 before a disjoint 45-task, 810-episode Cartesian product across three model families, clean/faulty execution, and three policies; it is not pooled with GOALCOMMITBENCH. Template-cluster bootstrap intervals and Holm-corrected sign tests use causal templates as the statistical unit. An append-only ledger retries only provider errors under the same logical identity; frozen analyzers reproduce headline results twice from fingerprint-bound artifacts (Appendix I).

Table 1: Headline matched Agent comparison (1,024 eight-model cells × 4 displayed policies = 4,096 episodes). Arrows mark preferred direction. Decl. cov. is the success-declaration rate and is an operating-point statistic interpreted jointly with declaration precision and DFs; Decl. prec.: correctness. Appendix 20 adds the strict no-commit control (0% declaration coverage) and reports escalation and call cost. DFs: dangerous false success. †VTC (Mansoor et al., 2026); ‡RAC (Perera et al., 2026); adapters are not official systems.

Method	Origin	DFs ↓	Harm ↓	Decl. cov.	Decl. prec. ↑	Closure ↑
Prompted-only	Control	9.4%	6.3%	83.1%	88.7%	74.8%
Verified-call adapter†	VTC-style	5.4%	9.0%	88.0%	93.9%	82.7%
Logged compensation‡	RAC/Saga-style	5.7%	11.2%	86.1%	93.4%	80.5%
GoalSeal-CG v3	Ours	**0.0%**	**4.7%**	82.2%	**100.0%**	**85.0%**

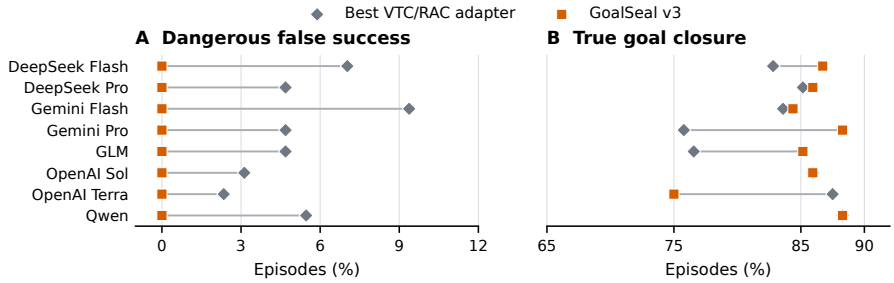


Figure 2: Matched sensitivity across eight models (128 cells per policy and model). Gray marks the stronger VTC/RAC-style adapter. GoalSeal v3 has zero dangerous false success in every stratum and highest or tied closure in seven of eight; strata are descriptive, not independent tests.

7 Results

7.1 Safety and autonomy under supplied contracts

Table 1 uses the exact eight-model intersection: every policy sees the same 1,024 task–fault–model cells and every semantic outcome is retained.

On this grid, GoalSeal-CG v3 combines 0% dangerous false success, 100% declaration precision, and the highest observed closure (85.0%) at 82.2% declaration coverage. VTC/RAC-style adapters have 5.4/5.7% danger (paired 95% template-cluster CIs for the reductions: [2.1, 9.4]/[3.2, 8.5]) and 82.7/80.5% closure. The 4.5-point closure gain over RAC/Saga has CI [0.6, 8.7]; the 2.2-point gain over VTC remains unresolved ([-2.0, 7.3]). More calls alone do not explain the gain: the strict certified control uses 15.9 calls versus v3's 15.2, yet closes 77.5% versus 85.0%. VTC/RAC use 11.2/11.6 calls, so those contrasts are quality–cost operating points, not equal-budget tests. Appendix 20 reports costs and Appendix 19 reports the mechanism track.

Declaration safety versus realized harm. All 48 harmful v3 episodes (4.7%) are `post_verify_revocation`: an external actor revokes state after an irreversible effect. All violate an invariant, but none is declared successful. Monitoring prevents false reporting, not realized harm.

Targeted v3 refinement: safety plus recovered autonomy. On a disjoint 45-task supportive retest (810 episodes), v3 has 0.0% faulty danger (0/135; one-sided 95% upper bound 2.2%; each model family 0/45, upper bound 6.4%). Clean closure rises 57.8%–84.4% (+26.7 points; 95% CI [14.8, 39.3]) and escalation falls 41.5%–14.1%. This prospective supportive retest passes all six pre-specified criteria.

Matched mechanism identification. Across 896 matched episodes, certified GoalSeal records 0.0% danger and 89.8% closure versus 12.5% and 85.2% for independent-evidence verify-before-retry (Holm $p = 0.000244$). These mechanism-equivalent implementations are not official author code; matched contrasts isolate independent evidence and the certificate.

Table 2: Service and cross-node validation. DFs use each track's registered in-bound or non-common fault set; common-mode controls are separate.

Track	N	Ctrl DFs ↓	GoalSeal DFs ↓	Resolution ↑	Boundary DFs ↓	Outage esc. ↑
Redis/HTTP witness	1,920	85.7%	0.0%	100.0% clean	100.0%	–
Cross-node stateful	300	78.6%	0.0%	100% close/contain	100.0%	–
Git–service composite	400	66.7%	0.0%	100.0% clean	–	100%
External-cloud replica	400	66.7%	0.0%	100.0% final	–	100%

7.2 Sensitivity to contracts and evidence

Contract and witness corroboration. Across 144 compiler calls and 1,152 downstream episodes, fixed $k = 2$ QuorumContract raises required-clause precision/recall from 80.7/70.3% to 93.5/100% and forbidden-group recall from 47.2% to 100%, removing a $+1.4$-point danger increase and narrowing the oracle closure gap from 16.0 to 6.2 points. Exact match remains 0%; omitted clauses remain outside the guarantee. In the 2,592-episode witness grid, the one-auditor fault bound yields zero danger and 75% closure (matching the oracle); common-mode false evidence outside that bound yields 11.1% danger.

8 Conclusion

Under explicit contract and fault envelopes, GoalSeal-CG v3 combines 0% observed danger, 100% declaration precision, and the highest observed closure (85.0%) across eight models. On disjoint ToolBench-X it passes 6/6 criteria with a 26.7-point clean-closure gain. QuorumContract reaches 100% required and forbidden-group recall without added danger. Omitted clauses, common-mode evidence, and external change remain limitations.

References

Laksh Advani. From confident closing to silent failure: Characterizing false success in LLM agents. *arXiv preprint arXiv:2606.09863*, 2026. doi: 10.48550/arXiv.2606.09863. URL https://arxiv.org/abs/2606.09863.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=33XGfHLtZg.

Tian Bai and Ying Jin. Conformal selective prediction with general risk control. *arXiv preprint arXiv:2603.24704*, 2026. doi: 10.48550/arXiv.2603.24704. URL https://arxiv.org/abs/2603.24704.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. doi: 10.48550/arXiv.2506.07982. URL https://arxiv.org/abs/2506.07982.

Hongliu Cao, Ilias Driouich, and Eoin Thomas. Beyond task completion: Revealing corrupt success in LLM agents through procedure-aware evaluation. *arXiv preprint arXiv:2603.03116*, 2026. doi: 10.48550/arXiv.2603.03116. URL https://arxiv.org/abs/2603.03116.

Arnold Cartagena and Ariane Teixeira. Mind the GAP: Text safety does not transfer to tool-call safety in LLM agents. *arXiv preprint arXiv:2602.16943*, 2026. doi: 10.48550/arXiv.2602.16943. URL https://arxiv.org/abs/2602.16943.

Ying Chen, Lihuang Fang, Rui Jiang, Mingxu Wang, Zhifeng Gu, Lei Yi, and Jie Chen. Done, but not sure: Disentangling world completion from self-termination in embodied agents. *arXiv preprint arXiv:2605.08747*, 2026a. doi: 10.48550/arXiv.2605.08747. URL https://arxiv.org/abs/2605.08747.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026b. doi: 10.48550/arXiv.2606.17573. URL https://arxiv.org/abs/2606.17573.

Liang Cheng, Mingsheng Cai, Jiuming Jiang, and Luo Mai. Do agents know what they can't do? evaluating feasibility awareness in tool-using agents. *arXiv preprint arXiv:2605.28532*, 2026. doi: 10.48550/arXiv.2605.28532. URL https://arxiv.org/abs/2605.28532.

C. J. Clopper and Egon S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404.

Pierre Dantas, Lucas Cordeiro, Ehsan Nowroozi, and Tihanyi Norbert. Toward safe LLM agents: A survey of specification, verification, and enforcement. *arXiv preprint arXiv:2608.14590*, 2026. doi: 10.48550/arXiv.2608.14590. URL https://arxiv.org/abs/2608.14590.

Youwang Deng. Goal-autopilot: A verifiable anti-fabrication firewall for unattended long-horizon agents. *arXiv preprint arXiv:2606.11688*, 2026. doi: 10.48550/arXiv.2606.11688. URL https://arxiv.org/abs/2606.11688.

Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for LLM agents. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: New Ideas and Emerging Results*, 2026. doi: 10.1145/3786582.3786839. URL https://doi.org/10.1145/3786582.3786839.

Sunny Dubey. Real-time detection and repair of LLM agent failures. *arXiv preprint arXiv:2608.02464*, 2026. doi: 10.48550/arXiv.2608.02464. URL https://arxiv.org/abs/2608.02464.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2151–2159. PMLR, 2019. URL https://proceedings.mlr.press/v97/geifman19a.html.

Takyoung Kim, Jinseok Nam, Chandrayee Basu, Xing Fan, Chengyuan Ma, Heng Ji, Gokhan Tur, and Dilek Hakkani-Tür. ReIn: Conversational error recovery with reasoning inception. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4J3kkHI6m5.

LangChain. Persistence. Software documentation, 2026. URL https://docs.langchain.com/oss/python/langgraph/persistence. Accessed 2026-08-10.

Dawei Li, Yuguang Yao, Zhen Tan, Huan Liu, and Ruocheng Guo. ToolPRMBench: Evaluating and advancing process reward models for tool-using agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12378–12391. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.602. URL https://aclanthology.org/2026.findings-acl.602/.

Bohao Liao, Jingchao Wang, Qipeng Song, Jin Cao, Jieling Wang, and Boyu Deng. TraceGrant: A contract-governed security framework for the task-effect lifecycle of networked LLM agents. *arXiv preprint arXiv:2608.21126*, 2026. doi: 10.48550/arXiv.2608.21126. URL https://arxiv.org/abs/2608.21126.

Yanming Liu, Xinyue Peng, Jiannan Cao, Xinyi Wang, Songhang Deng, Jintao Chen, Jianwei Yin, and Xuhong Zhang. ToolGate: Contract-grounded and verified tool execution for LLMs. *arXiv preprint arXiv:2601.04688*, 2026a. doi: 10.48550/arXiv.2601.04688. URL https://arxiv.org/abs/2601.04688.

Zheyuan Liu, Liqiang Xiao, Yang Li, Hyokun Yun, Lihong Li, Chao Zhang, and Meng Jiang. Do LLMs catch their own mistakes? a comprehensive benchmark for reflective tool use LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1748–1773. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.findings-acl.86. URL https://aclanthology.org/2026.findings-acl.86/.

Isham Kalappurackal Mansoor, Abhishek Phadke, and Pratip Rana. Verified tool calls improve LLM agent reliability under non-atomic failures. *arXiv preprint arXiv:2608.02645*, 2026. doi: 10.48550/arXiv.2608.02645. URL https://arxiv.org/abs/2608.02645.

Aritra Mazumder and Nusrat jahan Lia. AgentCheck: A reproduce-intervene-mitigate workbench for LLM agents over MCP. *arXiv preprint arXiv:2607.11098*, 2026. doi: 10.48550/arXiv.2607.11098. URL https://arxiv.org/abs/2607.11098.

Bardia Mohammadi, Nearchos Potamitis, Lars Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows. *arXiv preprint arXiv:2602.14849*, 2026. doi: 10.48550/arXiv.2602.14849. URL https://arxiv.org/abs/2602.14849.

Hai-Duong Nguyen and Xuan-The Tran. Verify-gated completion as admission control in a governed multi-agent runtime: A bounded architecture case study. *arXiv preprint arXiv:2605.17998*, 2026. doi: 10.48550/arXiv.2605.17998. URL https://arxiv.org/abs/2605.17998.

OpenAI. Openai agents sdk. Software documentation, 2026. URL https://openai.github.io/openai-agents-python/. Accessed 2026-08-10.

Srinath Perera, Kaviru Hapuarachchi, Frank Leymann, and Rania Khalaf. Robust agent compensation (RAC): Teaching AI agents to compensate. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 253–262, 2026. doi: 10.1145/3786335.3813141. URL https://arxiv.org/abs/2605.03409.

Ella Rabinovich, David Boaz, Naama Zwerdling, and Ateret Anaby Tavor. Near-miss: Latent policy failure detection in agentic workflows. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 296–308. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.gem-main.30. URL https://aclanthology.org/2026.gem-main.30/.

Vikas Reddy, Sumanth Reddy Challaram, and Abhishek Basu. Reason less, verify more: Deterministic gates recover a silent policy-violation failure mode in tool-using LLM agents. *arXiv preprint arXiv:2607.07405*, 2026. doi: 10.48550/arXiv.2607.07405. URL https://arxiv.org/abs/2607.07405.

Junhao Su, Yuanliang Wan, Junwei Yang, Hengyu Shi, Tianyang Han, Yurui Qiu, and Junfeng Luo. Failure makes the agent stronger: Enhancing accuracy through structured reflection for reliable tool interactions. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12712–12734. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.618. URL https://aclanthology.org/2026.findings-acl.618/.

Jimin Sun, So Yeon Min, Yingshan Chang, and Yonatan Bisk. Tools fail: Detecting silent errors in faulty tools. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14272–14289. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.790. URL https://aclanthology.org/2024.emnlp-main.790/.

Linzhuang Sun, Mingyang Chen, Hao Liang, Tianpeng Li, Zhou Yijie, Chenzheng Zhu, Tianyu Guo, Huanyao Zhang, Jingxuan Wei, Bihui Yu, Fan Yang, and Wentao Zhang. Training verifier to assessing complex real-world tool-use trajectories. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32922–32936. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1647. URL https://aclanthology.org/2026.findings-acl.1647/.

Temporal Technologies. Temporal platform documentation. Software documentation, 2026. URL https://docs.temporal.io/. Accessed 2026-08-10.

Yang Tian, Zhengpeng Shi, Yu Zhou, and Bo Zhao. Beyond function calling: Benchmarking tool-using agents under tool-environment unreliability. *arXiv preprint arXiv:2606.25819*, 2026. doi: 10.48550/arXiv.2606.25819. URL https://arxiv.org/abs/2606.25819.

Eduardo Treviño, Hugo Contant, James Ngai, Graham Neubig, and Zora Zhiruo Wang. Benchmarking failures in tool-augmented language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2916–2934. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.149. URL https://aclanthology.org/2025.naacl-long.149/.

Jiaxuan Wang, Yulan Hu, Wenjin Yang, Zheng Pan, Xin Li, and Lan-Zhe Guo. Aligning agents via planning: A benchmark for trajectory-level reward modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 23174–23200. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1062. URL https://aclanthology.org/2026.acl-long.1062/.

Cailin Winston, Claris Winston, and René Just. Solver-aided verification of policy compliance in tool-augmented LLM agents. *arXiv preprint arXiv:2603.20449*, 2026. doi: 10.48550/arXiv.2603.20449. URL https://arxiv.org/abs/2603.20449.

Ke Yang, Panpan Li, Zonghan Wu, Kejin Xu, Huaxi Huang, and Xiaoshui Huang. DART: Semantic recoverability for structured tool agents. *arXiv preprint arXiv:2605.23311*, 2026. doi: 10.48550/arXiv.2605.23311. URL https://arxiv.org/abs/2605.23311.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Xiaoli Yu and Jiamiao Liu. A joint finite-sample certificate for adaptive selective conformal risk control. *arXiv preprint arXiv:2606.08517*, 2026. doi: 10.48550/arXiv.2606.08517. URL https://arxiv.org/abs/2606.08517.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Zhiwei Zhang, Fei Zhao, Rui Wang, Zezhong Wang, Bin Liang, Jiakang Wang, Yao Hu, Shaosheng Cao, and Kam-Fai Wong. Robust tool use via fission-GRPO: Learning to recover from execution errors. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 40477–40491. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1880. URL https://aclanthology.org/2026.acl-long.1880/.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. FLARE: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22391–22401, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhao_FLARE_A_Failure-Aware_Framework_for_Autonomous_Correction_and_Recovery_in_CVPR_2026_paper.html.

Dongsheng Zhu, Xuchen Ma, Yucheng Shen, Xiang Li, Yukun Zhao, Shuaiqiang Wang, Lingyong Yan, and Dawei Yin. When tools fail: Benchmarking dynamic replanning and anomaly recovery in LLM agents. *arXiv preprint arXiv:2606.05806*, 2026. doi: 10.48550/arXiv.2606.05806. URL https://arxiv.org/abs/2606.05806.

AI USE STATEMENT

Generative AI tools assisted literature discovery and synthesis; research-question and hypothesis refinement; conceptual and mathematical formulation; proof drafting and critique; synthetic benchmark construction; experimental-design review; method, evaluation, and analysis-code implementation; test generation; statistical interpretation; and manuscript drafting and editing. The authors manually checked the cited literature and mathematical arguments, inspected the frozen protocols and generated artifacts, executed the reported experiments, and admitted empirical values only through deterministic fingerprint-bound analyzers and independent reproduction checks. The authors remain responsible for all methodology, code, claims, citations, analyses, and text, including content produced with generative-AI assistance.

A Contract and Runtime Details

Each contract is serialized as typed JSON. Required and forbidden clauses are executable predicates over simulator state; invariants additionally receive the complete prefix trace. The runtime never exposes evaluator truth to the language model. It sees only documented tool outputs and the observations that its policy requests. The evaluator latches invariant violations even if later compensation restores a valid terminal state.

For every intended effect, the ledger stores the requested resource, expected state delta, acknowledgement, last observed version, idempotency key, reversibility class, compensation handle, and whether execution occurred. A declaration is terminal: the model must explicitly choose `commit` or `escalate`. The runtime may reject an inadmissible declaration and return violated clauses, remaining counterexample worlds, informative probes, and harm-admissible interventions.

The finite-world backend stores for each world a closure label, violated clauses, fault labels, and weight. Each probe maps every modeled world to an end-to-end outcome and records cost, latency, harm, and failure domain. Each intervention maps every world to a successor and records per-world harm. A modeled intervention carries the possible-world set through its transition relation; the end-to-end LLM adapter conservatively regenerates the full task model after every mutation and then conditions it on fresh observations. Exact witness search supports small probe sets; greedy cover and adaptive pair-mass elimination provide scalable alternatives. Observations outside modeled support fail closed.

For the executable benchmark adapter, every declared action independently admits normal, silent no-op, partial, wrong, duplicate-increment, and documented compensation realizations. The Cartesian state construction is exhaustive for the catalog's declared action order (at most four actions). Each semantic state induces a truthful observation world and, where distinct, a primary-stale world. For every modeled closed/open state pair we additionally construct both asymmetric revocation worlds: primary appears closed while the independent ledger is open, and vice versa. Therefore a single success-shaped observation cannot eliminate all open worlds in either probe order. Transport-only version counters are excluded from semantic outcomes; task predicates, forbidden groups, and invariant groups determine the closure label. The runtime separately latches any invariant violation observed earlier in the trace.

B Decision Objects of Concurrent Completion Gates

The closest systems share a fail-closed intuition but certify different objects. Table 3 states the conceptual comparison; Table 4 separately reports a post-unblinding same-interface experiment. The latter is a protocol-port comparison, not a claim about the authors' full systems.

Table 3: Decision-object comparison with the closest concurrent completion gates.

System	Admitted object	Evidence semantics	Recovery admission
Goal-Autopilot (Deng, 2026)	Terminal `DONE` after the gates on an accepting FSM path execute and pass	Boolean executable state gates; the theorem is conditional on gate soundness, floor enforcement, and plan coverage	Per-state known fixes and bounded retries; otherwise an honest stall
Verify-gated completion (Nguyen & Tran, 2026)	A proposed completion claim admitted by a read-only verifier	Versioned claim, evidence, common-ground, ownership, and freshness packets	A named recovery packet and owner re-enter verification; rollback is human reviewed
GoalSeal-CG (ours)	Task-level commit only when every world compatible with the transcript is goal-closed	Probe outcomes condition a belief over state, transition, provenance, freshness, and common-cause fault variables	A mutation is admitted only if its transition is harm-safe in every remaining world; otherwise probe or escalate

Table 4: Same-benchmark nearest-gate comparison over 128 paired conditions (384 episodes). This post-unblinding extension is descriptive; ports are not the authors' full systems.

Method	DFs ↓	Closure ↑	Esc. ↓	Harm ↓	Calls ↓
Goal-Autopilot port	0.0%	84.4%	15.6%	4.7%	9.0
Verify-gated port	10.9%	37.5%	51.6%	12.5%	4.4
GoalSeal	0.0%	89.8%	10.2%	4.7%	16.7

Nearest-neighbor port audit. The Goal-Autopilot port is based on the public author artifact at commit `887950f5a4783dd5bf54fa91c26bc51c17fe6ee7` (MIT license). It preserves four terminal-admission invariants: executable per-state gates, an explicit plan-coverage audit, bounded known repair, and a hard `DONE` floor. The supplied GoalCommitBench contract is compiled into the definition-of-done gate and action postconditions into state gates; both it and GoalSeal execute gates against the same independent observation channel. It does not reproduce the authors' LLM compiler, prompts, shell scheduler, or SWE-bench environment and is therefore labeled *protocol port from author code*, not author code. The verify-gated row preserves proposal–verifier separation, read-only admission, packetized evidence, and fail-closed rejection; no official benchmark runtime is claimed by our adapter. VIGIL is an evaluation protocol, not an action policy, so we use its world-completion/terminal-commitment decomposition in our metrics rather than fabricate a runtime row. The complete mapping and executable invariant tests are released in `config/nearest_gate_adapter_fidelity_v1.md`.

Table 5: Implementation provenance for empirical baseline families. A method name in the paper never upgrades an adapter's provenance.

Family	Provenance	Preserved surface	Validation / limitation
VTC	Faithful public-algorithm reimplementation	Three-valued postcondition, verify-before-retry, stable idempotency key, single retry	Unit conformance; no author code linked by source paper
RAC	Author component + separate mechanism adapter	Author RecoveryManager effect log/compensation; matched row uses Saga-style reverse compensation	Native smoke at frozen commit; matched table row is not author code
DART	Faithful definition-level reimplementation	Recovery localization and semantic admissibility predicates	Paper-linked repository implements a different guard surface; deviation retained in registry
Goal-Autopilot	Protocol port from public author artifact	State gates, A3 coverage audit, bounded repair, hard terminal floor	Commit-pinned invariants; excludes compiler, prompts, scheduler, and original task environment
Verify-gated	Architecture-equivalent	Read-only admission, fail-closed packet decision	No official task benchmark runtime claimed; excludes recovery effectiveness
Temporal / LangGraph / Agents SDK	Native-component adapters	Durable replay, checkpoint/interrupt, or tool guardrail primitives	Version-pinned native smokes; not production leaderboards

Assurance labels are premise-sensitive. An executable certificate records three fields separately: the closure verdict, the contract provenance/coverage status, and the modeled fault/evidence envelope. "Model-relative certified" is permitted only when all required clauses have an accountable source and the claimed fault bound covers every evidence group used by the proof. If clause coverage is unknown, the runtime may expose a bounded diagnostic but not a certified completion. If a common cause is declared but lacks a distinguishing witness, Proposition 1 requires escalation. No label can detect an omitted world that is observationally identical to a modeled closed world; the purpose of the label is to make that premise auditable rather than implicit.

C Contribution–Evidence–Scope Map

The technical components of our three contributions form one dependency chain: the admission theorem assumes a goal/world model; the runtime operationalizes that rule; the benchmark tests its safety–autonomy behavior under controlled ambiguity; and the contract track measures and mit-

igates failure of the theorem's semantic premise. Table 6 binds each claim to its evidence and inference boundary rather than treating artifact size as evidence by itself.

Table 6: Claim–evidence map. "Not claimed" entries are part of the claim boundary, not deferred positive conclusions.

Contribution	Direct evidence	Inferential unit	Not claimed
Closure identification	Propositions 1–4, complete appendix proofs, exact witness oracle, and the 1,080-cell common-mode envelope	Deterministic finite model; no stochastic sample for the theorems	Distribution-free safety, correctness under an omitted world or goal clause, or production fault prevalence
GOALSEAL-CG runtime	3,456 Agent episodes, the complete 896-episode seven-policy mechanism track, 1,080 held-out ToolBench-X episodes, 2,592 dual-auditor episodes, the 300-episode stateful cross-node repair study, and exact common-mode controls	Causal template for the main grid; task cluster for ToolBench-X sensitivity; registered capability scenario for witness quorum; logical episode for the stateful grid	Exactly-once execution, general rollback, superiority on every autonomy target, real-service fault prevalence, safety outside the one-auditor bound, or immunity to common-mode false evidence
Benchmark	Sixteen sealed hidden causal templates, state-truth outcomes, matched modes, hidden fault compositions, and disjoint public-tool tasks	Independently designed causal template; parameterized trajectories remain dependent	A prevalence estimate for real failures or evidence that synthetic service boundaries are organizationally independent
Contract premise and QUORUMCONTRACT	144 frozen compiler calls and 1,152 compiled-contract downstream episodes	Contract/task template	Truth from model agreement or correctness under a jointly omitted required clause

D PROOFS AND WITNESS-SELECTION DETAILS

Proof of Proposition 1. Fix an allowed observation policy and let T be a transcript on which the monitor commits with probability $q > 0$. Observational indistinguishability gives the same distribution over T in w^+ and w^-. Hence the monitor's conditional action distribution after T is also the same in both worlds. In particular, it commits with probability q in the goal-open world w^-, producing dangerous false success with positive probability. Randomized prompting or a confidence score only changes q and cannot make the two worlds distinguishable. □

Set-cover equivalence and proof of Proposition 2. Let $S_p \subseteq U$ contain the pairs distinguished by probe p. A probe protocol is a valid non-adaptive witness exactly when its sets cover U, and its objective is exactly $\sum_p c(p)$. Thus minimum witness selection is weighted set cover. Conversely, any weighted set-cover instance can be represented by one closed–open pair per universe element and one binary probe per input set whose outcomes differ exactly on the represented elements, proving NP-hardness.

For the approximation claim, consider a greedy iteration with r pairs uncovered. An optimal cover of total cost OPT covers these r residual pairs, so at least one of its probes has cost per newly covered pair at most OPT/r; otherwise summing newly assigned coverage over that cover would cost more than OPT. The implemented greedy rule has no larger ratio. Charge its selected probe's cost equally to the pairs it newly covers. As pairs are charged, the residual count decreases from $|U|$ to zero, and the charge to the pair removed when r pairs remain is at most OPT/r. Hence total greedy cost is at most $\mathrm{OPT}\sum_{r=1}^{|U|} 1/r = H_{|U|}\,\mathrm{OPT} \le (1+\ln|U|)\,\mathrm{OPT}$. Positive probe costs make every ratio well-defined. □ The adaptive rule is evaluated empirically against the exact non-adaptive oracle; this proof makes no guarantee for its contingent decision tree.

Proof of Proposition 3. Under the stated model-coverage premise, the realized world belongs to the initial belief set. Every action transition and observation-conditioning step preserves membership of the realized successor because its transition and emitted observation are contained in M. Inductively, the realized world therefore remains in $\mathcal{B}_M(\tau)$ at every decision point. If the monitor commits only when every member of this set satisfies C_G, the realized world satisfies C_G as well, so dangerous false success is impossible relative to (G, M). □

Proposition 4 (maximal sound admission). For fixed (G, M), define $a^*(\tau) = \mathbf{1}[\mathcal{B}_M(\tau) \subseteq \{w : C_G(w) = 1\}]$. Among all admission rules measurable from the same transcript, a^* is the unique pointwise-maximal model-relative sound rule: every sound rule a satisfies $a(\tau) \leq a^*(\tau)$ for every reachable τ. Thus sound coverage outside this region requires distinguishing evidence, safe intervention, or a stronger declared model rather than rescoring the unchanged transcript. An implementation attains the bound only if world construction and counterexample elimination are complete for (G, M); our certificate audit checks admission consistency, not that completeness.

Proof. First, a^* is sound by Proposition 3. Now take any reachable transcript τ with $a^*(\tau) = 0$. By definition there exists $w^- \in \mathcal{B}_M(\tau)$ such that $C_G(w^-) = 0$. Any rule measurable from the same transcript has the same action on every world compatible with τ; therefore, if it commits at τ, it also commits in w^- and is not model-relative sound. Consequently every sound deterministic rule has $a(\tau) = 0$ whenever $a^*(\tau) = 0$, while it can be at most one where $a^*(\tau) = 1$. Hence $a \leq a^*$ pointwise, and including every transcript admitted by a^* makes it uniquely maximal. For a randomized rule, replace $a(\tau)$ by its conditional commit probability: soundness forces that probability to zero wherever an open compatible world exists. □

Finite-world scaling diagnostic. Before measuring runtime, we froze the generator, analyzer, seed, five exact points, five scale points, and five timed repeats per method. The deterministic instances instantiate the set-cover reduction with one closed world, a planted feasible cover, and distractor probes. Exact enumeration grows from 0.97,ms at 33 worlds/eight probes to 1.11,s at 513 worlds/16 probes. Greedy takes 3.98,ms at the latter point, with observed cost/optimum ratios from 1.37 to 1.79 across the five exact instances. On the largest scale point (16,385 worlds and 128 probes), greedy takes 1.77,s and 42.99,MiB of traced Python allocations; the closed-path adaptive selector takes 5.17,s and 2.57,MiB. All values are medians of five runs on one machine. This diagnostic establishes executable scaling within the registered finite-world regime, not production latency, cross-hardware performance, or tractability for unbounded state spaces.

E BENCHMARK CONSTRUCTION AND SPLITS

Table 7: GOALCOMMITBENCH composition. "Instances" are executable surface variants; the independently designed causal template is the inferential unit.

Split	Templates	Instances	Role
Calibration	8	80	Schema, adapter, and generator checks
Development	8	80	Policy design, audits, and operating-point selection
Hidden test	16	160	Structurally held-out confirmatory evaluation
Total	32	320	Four balanced domains; ten variants per template

Catalog and record schema. The catalog contains 32 authored causal workflow templates and ten executable surface instances per template. It is balanced across travel, retail, calendar/payment, and shipping: each domain contributes two calibration, two development, and four hidden templates. An instance changes entities, amounts, dates, resource identifiers, and whether the irreversible action advertises an idempotency key; it does not change the template's state-transition or causal fault-graph structure. Each task record contains (i) an instruction and initial state, (ii) a typed action sequence with normal, partial, wrong, and compensation effects when applicable, (iii) required terminal predicates and full-trace invariants, and (iv) metadata binding its domain, template, split, and instance. Every irreversible action is non-idempotent; idempotency-key support is balanced over surface instances.

From tasks to closure cases. For a task, the world compiler enumerates reachable semantic states induced by the registered action outcomes. Each semantic state receives a truthful observation world and, when distinct, a stale-primary world. Closed/open state pairs additionally receive asymmetric post-verification revocation worlds, so reversing probe order cannot by itself make correlated evidence independent. A case is retained only when its weak-protocol transcript is compatible with both a closed and an open world and the registered probe set contains a distinguishing observation.

Probes record outcome by world, cost, latency, failure domain, and freshness; interventions record their successor and harm in every world. The exact solver supplies the minimum non-adaptive witness cost used for verification-regret scoring. Observations outside the executable support fail closed instead of deleting all counterexamples.

Fault families. Action faults include explicit failure, silent no-op, partial commit, wrong write, and duplicate effect. Observation faults include acknowledgement loss, delayed visibility, and stale primary or shared-cache reads. Environment faults include external mutation and post-verification revocation. Recovery faults include failed compensation and compensation that returns a success-shaped acknowledgement without restoring state. Single and compound schedules are instantiated only where their semantics apply: compensation failures require a compensable action, and revocations require an observable prior effect.

Structural splitting and schedule binding. Workflow templates are split by state-transition and causal fault-graph motif, not by lexical similarity. The registered catalog has 8 calibration, 8 development, and 16 hidden templates, corresponding to 80, 80, and 160 surface instances. No causal motif crosses a split. The hidden catalog is constructed deterministically and is inaccessible to ordinary development runner invocations. The main confirmatory run uses one deterministically selected instance per hidden template because the ten surface variants are dependent realizations, not independent evidence.

Fault locations are selected with a stable SHA-256 mapping from experiment seed, task identifier, and fault scenario before models or runtime modes are crossed. The condition manifest is therefore invariant to model-configuration and policy order. Its exact product contains 16 hidden templates, eight fault schedules, nine model configurations, and three policies, yielding 3,456 episodes over 1,152 matched condition–model triples. Statistical uncertainty resamples templates with all nested variants rather than treating those rows as independent.

Validation boundaries. Deterministic validators check schema typing, predicate keys, action reachability, closed/open membership, nonempty distinguishing evidence, transition support, and exact schedule cardinality. The separate 24-item blinded human audit samples three instances from every development template and checks required and invariant clauses from instructions and advertised tool effects. Table 18 reports its finite-sample agreement and limitations. It does not human-validate the sealed hidden templates, estimate production fault prevalence, or establish that the executable world model covers failures outside the registered ontology.

F Frozen Confirmatory Protocol

Conditions are paired by model, task, realized world, fault position, and seed. The historical v9 protocol disabled automatic provider retries; the versioned v10 completion policy below retries only transport/provider failures and retains every attempt. Decoding parameters are frozen per route: temperature zero is requested where supported, while routes that reject temperature control use and report the provider default. Interrupted runs resume only if their full fingerprint matches; the fingerprint includes model routes, prompts, per-task tool schemas, selected causal templates, possible-world and probe semantics, condition manifests, runtime version, baseline configuration, episode schema, analysis protocol, and route-level decoding controls. The primary closure field requires terminal clauses, absence of terminal forbidden effects, and no latched execution-time invariant violation.

Operational quota windows may split one frozen run into ordered batches without changing its scientific fingerprint or schedule order. Each batch records its start, size, completion count, and end reason. Eight consecutive episode-level provider failures trigger a resumable circuit break. Independently, a complete run is ineligible for inference if its API-error episode rate exceeds 5% or its longest consecutive API-error block exceeds seven; an invalid run is retained in full and never repaired by selectively replacing failed rows. This infrastructure rule was amended outcome-blind on 2026-08-21 after a six-error circuit break and before inspecting any Agent efficacy field; the earlier checkpoint remains a historical protocol-v4 run and is not retroactively reclassified as protocol v5. A development run that completed all 960 identities but suffered 840 API-error episodes, including a terminal block of 810, is therefore retained only as an infrastructure audit and excluded from every

efficacy table. A subsequent batched run was likewise archived when its circuit breaker stopped execution at 50/960 after the first six-error block. A later 19/20 connectivity pass was immediately followed by a fresh run with only two normal episodes and then six consecutive cross-route errors, stopping at 8/960. Every new run therefore requires three complete connectivity reports at least five minutes apart, with no transport or route failure. All semantic probe failures remain compatibility outcomes but do not condition run opening, avoiding selection for a favorable stochastic model response. This scope was amended before any infrastructure-eligible pilot outcome was observed. Both live runners reject a closed, malformed, future-dated, or stale gate report and bind its digest into each batch record. A later v1 checkpoint reached 90/960 without API errors but remained incomplete and was never efficacy-analyzed. When the platform supplied a native Anthropic Vertex SSE endpoint, we advanced the adapter to v2, fingerprinted the API format, archived the partial v1 checkpoint, and required a from-zero v2 run. The native endpoint still reported a different model identity for the requested Fable route, so no v2 efficacy episode was opened. Before any v2 efficacy or hidden result was opened, we prospectively removed that route rather than substitute an unvalidated model. The revised nine-configuration schedules contain 864 development and 3,456 hidden Agent episodes; the automatic-contract schedules contain 72 development and 144 hidden model calls. Anthropic-family coverage remains through the Sonnet configuration. All four Agent/contract candidate/live fingerprints were regenerated, while historical ten-model infrastructure artifacts retain their original labels and counts.

After repeated provider failures under that zero-retry protocol, we versioned the final Agent experiment as a case-completion estimand rather than silently repairing v9. The frozen 3,456 logical identities and their order are unchanged, but provider executions are stored in an append-only attempt ledger. Only transport or provider API errors are retried; the first non-API-error attempt completes a logical case, and any ordinary task failure in that attempt remains the outcome. Thus retrying cannot select a more favorable Agent answer. The v10 track has neither an API-error-rate exclusion nor a consecutive-error circuit breaker. It instead reports total and failed attempts, retried-case incidence, maximum attempts, latency, and token overhead as separate reliability and cost outcomes. The 1,122 attempts in the preserved v9 checkpoint were imported without alteration: 1,112 already completed logical cases and ten API-error attempts requiring a second execution. An independent completion auditor reconstructs the schedule, checks contiguous per-case attempt indices and the first-clean projection, and prohibits efficacy analysis until all 3,456 logical cases and the final seal are present. The historical v9 artifacts remain excluded under their own registered gates and are not reclassified as valid v10 runs.

While the hidden Agent checkpoint was incomplete, an infrastructure-only auditor verified its exact schedule prefix and explicitly reported that no efficacy field had been inspected. We then sealed the already registered task-cluster, leave-one-template-out, and model-stratified sensitivity outputs, together with all registered cost, coverage, latency, failure, token, and route diagnostics. A public analysis-freeze record binds the analyzer, analysis core, experiment protocol, and checkpoint-auditor hashes; the submission verifier rejects source drift. A separately frozen completion-only audit additionally requires every execution batch to be closed and the final ledger to terminate as `complete`, so a full row count alone cannot open the analysis. Before opening hidden efficacy, we also froze the deterministic five-track result exporter, its six-macro manuscript interface, and a non-selective interpretation policy that reports every registered primary outcome regardless of direction or significance. This completes the preregistered reporting surface without changing an outcome, contrast, multiplicity family, or hidden schedule.

Before hidden opening, we additionally froze a deterministic mechanism track with seven policies, the same eight registered fault schedules, and one task per causal template. The final development run contains 448 episodes over 64 paired conditions; the hidden schedule contains 896 episodes over 128 paired conditions. Its complete proposed policy is certified GoalSeal. Primary comparators are an independent-evidence verify-before-retry/idempotency wrapper and a logged-compensation recovery policy; both are labeled mechanism-equivalent. The former has the same observation channel as the proposed policy but no certificate or irreversible-action gate, controlling evidence access. A same-source wrapper is descriptive. A first development artifact was rejected whole because its metadata encoded structured policy records where the strict analyzer required policy-name strings. A complete rerun was retained but superseded when the analysis fingerprint was expanded to bind the comparison family. An adversarial pre-hidden methods review then identified evidence-access confounding, so we added the matched control and executed another complete development

run. The frozen v3 fingerprint covers the runner, analysis version, exact condition manifest, policy/provenance manifest, calibration fingerprint, and risk budget. No hidden efficacy episode was executed during any correction.

The development artifact contains 20 structurally unique finite-world cases and is used only to test exact/greedy witness selection, out-of-support behavior, unsafe-retry rejection, and baseline adapters. A separate runtime gate crosses all 80 development tasks with all 18 registered single and compound fault schedules (1,440 episodes). Its first pass exposed unsupported default partial/wrong writes, temporal stale-probe rounds, and unsafe projected non-irreversible transitions. After extending the world compiler and mutation gate, the unchanged gate produced zero dangerous false successes, zero controllable irreversible harms, zero out-of-support observations, and zero API/protocol failures. Sixteen harms remained, all caused by an injected post-verification external revocation after an irreversible effect; these are retained as detected but already-realized harm, not relabeled as success. Neither development gate is confirmatory evidence. Before any hidden result was opened, a prospective power amendment expanded the hidden split from eight to sixteen causal templates without changing the eight development templates or their live schedule. The added templates exercise fan-out/join, an irreversible middle action, split settlement, identity binding, exception synchronization, quorum gating, multi-resource synchronization, and post-mutation settlement. Hidden fault compositions and external environments test model misspecification; any residual statistical calibration is performed at the causal-template level.

G Constructed Model-Misspecification Boundary

We separately construct a falsification-oriented development diagnostic without changing any frozen hidden schedule. One task from each of eight development templates generates 1,072 truth cases and 4,288 monitor–case rows across four monitor specifications and five hazard labels. The current method pair model commits in all constructed common-cause false-closed and post-pair-revocation cases: the primary and independent observations are identical to those of a modeled closed world. Adding observationally identical open twins to M reduces dangerous false success to zero on those cases but forces escalation on every nominal closed case when only that pair is observable. A modeled authoritative witness restores correct commit on every nominal case while retaining zero dangerous false success on the constructed common-cause cases. These rows are exhaustive constructions over clause signatures, not independent samples; the hazard variants share observation structure, no inferential test is performed, and the experiment neither estimates real-world fault prevalence nor assumes that such an authoritative witness exists in deployment.

A separate frozen reference-plan replay produced 1,656 matched episodes on 69 eligible τ^2-bench tasks. Across faulty cells, oracle GoalSeal-CG closed 100% of goals versus 33.0% for acknowledgement, 55.7% for retry, and 92.5% for same-source verification (paired task-cluster difference: 5.2–9.9 points); an exact rerun reproduced every row. Its reference plan, exact state hashes, independent truth, and lossless checkpoints make this executor evidence rather than autonomous-Agent evidence.

A disjoint prospective track compares upstream-no-hint with the executable `goalseal_runtime_v1` gate on the same tasks and crossing. It starts only after both the Agent grid and its terminal execution ledger are completion-sealed, uses six ordered 20-cell no-retry batches, and requires a fresh three-probe route certificate per batch. The full/batch drivers, probe and stability checker, model configuration, and connectivity protocol are prospectively hash-bound. Admission requires the exact manifest and resume order, clean/faulty injection checks, terminal ledger, route identity, and infrastructure thresholds to pass outcome-blind audits. Only then does the sealed exporter generate efficacy text. This one-route track tests independently authored executable-runtime validity, not broad model generalization.

The selective-runtime v2 extension addresses that breadth and false-escalation limitation with a second prospective manifest. Its 60 test tasks are disjoint from both the 30-task development pool and an invalidated first test manifest; invalidation occurred when formal stdout exposed an outcome before the aggregate-analysis boundary, and none of those rows is reused. The held-out Cartesian product crosses OpenAI Terra, Gemini Flash, and GLM routes with clean/faulty execution and upstream-no-hint, GoalSeal v1, and GoalSeal v2 modes, for exactly 1,080 logical episodes. Claude was removed outcome-blind after its multi-turn tool-continuation route returned systematic HTTP

404 responses; Qwen and DeepSeek passed development compatibility smokes but were excluded before the test because a single case required more than 30 model responses. These infrastructure dispositions are not efficacy exclusions. The selected auxiliary threshold is zero: this removes only an additional confidence cutoff, while missing calls, invocation failures, negative inner status, missing required fields, and ungrounded finals remain non-waivable. Provider attempts are retained and only provider/transport errors may be retried; ordinary task failures are never replaced. The frozen analysis reports every registered criterion, model stratum, task-cluster interval, multiplicity-adjusted comparison, and provider-retry count.

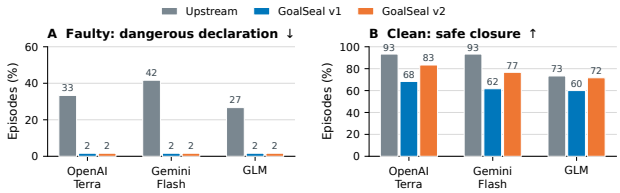


Figure 3: Held-out selective-runtime rates by model family; points are paired by task and execution condition.

Table 8: Post-hoc localization of the held-out ToolBench-X residual. The registered exact-match outcome is retained; PP denotes the effective-injection per-protocol sensitivity.

Audit target	Observation	Interpretation
Injection noncompliance	27/540 assigned-fault cells (3 tasks)	0 dangerous rows
V2 registered residual	3/180 ITT; 3/171 PP	one task; all 3 model families
Residual injection status	3/3 effectively injected	2, 2, and 3 effective mutations
Terminal representation audit	3/3 embed exact reference value	typed wrapper serialized as final text
Full-grid representation flags	9/1,080; one faulty task	3 modes $\times$ 3 model families
Typed-value sensitivity	danger: 58/180 upstream; 0/180 v1/v2	primary rows retained
Paired-effect sensitivity	-32.2 pp danger; $+4.4$ pp faulty safe	exactly invariant
Registered criteria	4/6 ITT; 4/6 PP	same two clean-autonomy misses

This classification is deliberately conservative. Across the full 1,080-row grid, exactly nine dangerous flags share the same faulty task—one for each runtime mode and model family. In every row, the policy candidate or value embedded by the typed terminal wrapper equals the reference, but the adapter forwards the wrapper representation to the strict string matcher. Typed-value normalization would yield faulty danger counts of 58/180, 0/180, and 0/180 for upstream, v1, and v2, while leaving the v2–upstream danger and v2–v1 faulty-safe paired contrasts exactly invariant. None of the 27 injection-noncompliant cells is dangerous. We therefore classify the cluster as a shared representation-boundary mismatch, not three independent model failures or an injection-compliance artifact. Because this is a post-hoc representation audit, it does not replace the registered exact-match outcome, repair a test row, or turn the 1.7% primary rate into zero.

After completing that registered analysis, we tested a stricter lineage-replay waiver on a new 30-task development pool with zero overlap with 150 prior cases. Separate v1 and v2 collections each completed the 60 clean/faulty episodes with no terminal provider error. A truth-blind lineage screen found only three soft-only candidates, but an authoritative-root audit admitted none: the invoice traces depended on unattested fallback records and the permit trace on unattested registration-state reads. Consequently the strict rule added no waiver; v2 improved clean safe closure over v1 by 6.7 points on this pool, not the targeted 20. We rejected the mechanism before freezing a confirmatory test. A post-hoc certificate audit also constructs a model-relative closed/open pair for every residual v2 escalation by varying an explicitly unsupported, failed-source, or otherwise unobserved obligation while holding the recorded observation fixed. This construction is an admission-consistency diagnostic, not an empirical efficacy result or proof that the declared fault model is complete. It explains why hashing or replaying a terminal calculation cannot certify source truth and why we do not relax the reported gate to improve coverage.

Targeted v3 gate and a prospective supportive retest. A development analysis of the v2 residual attributes the remaining clean over-escalation to a single over-restriction: v2 waives a soft (auxiliary) gap only through a terminal producer carrying *no* gap, so a producer that succeeded but reports a benign, non-required `unsupported_fields` entry cannot ground the final state. The candidate v3 gate relaxes exactly this condition—a succeeded terminal producer with only soft gaps grounds the waiver, while every hard gap (missing/failed call, negative inner status, missing required field, ungrounded final) stays fail-closed—and its auxiliary threshold is re-frozen at zero on development data before any test episode. On a newly frozen prospective manifest of 45 tasks (three per workflow–hazard stratum) disjoint from the development pool and both prior test manifests, an outcome-blind freeze then runs the same $45 \times 3 \times 2 \times 3 = 810$-episode grid. The exact pre-run freeze binds the runner, gate, manifest, episode schedule, thresholds, and six decision criteria. However, this executed grid differs from the preceding 60-task/1,080-episode preregistration, and the actual v3 analysis implementation was not prospectively hash-bound (the freeze referenced the preceding v2 analyzer). We therefore classify this as a prospectively specified supportive retest, not exact preregistration compliance. Within that status, v3 meets both pre-specified clean-autonomy criteria on the point estimate (clean safe closure $+26.7$ points, 95% task-cluster CI $[14.8, 39.3]$, Holm $p = 3.0\text{e}{-9}$; clean escalation -27.4 points, CI $[-40.0, -15.6]$, Holm $p = 2.4\text{e}{-9}$) while retaining 0% faulty danger, and all six criteria pass in both intention-to-treat and effective-injection per-protocol analyses. Both clean-autonomy effects are significantly positive (intervals exclude zero), but their 95% intervals extend below the 20-point bar, so we claim the pre-specified criteria are met, not that either effect provably exceeds 20 points. This is a candidate refinement on a new, smaller task set: it does not overwrite the frozen four-of-six v2 result on its own 60-task manifest, and it does not address the automatic-contract exactness bottleneck.

Table 9: Prospective v3 supportive retest (45 tasks, 810 episodes). The decision criteria were fixed before execution, but the 45-task run deviates from the preregistered 60-task sample size and is not presented as exact preregistration compliance.

Environment	Policy	DFs ↓	Safe closure ↑	Esc. ↓
Clean	Upstream	2.2	86.7	11.1
Clean	GoalSeal v1	0.7	57.8	41.5
Clean	GoalSeal v3	1.5	84.4	14.1
Faulty	Upstream	37.8	28.1	34.1
Faulty	GoalSeal v1	0.0	11.1	88.9
Faulty	GoalSeal v3	0.0	22.2	77.8

Prospectively frozen capability-isolated confirmation. An earlier 1,440-row candidate is retained but invalidated because its repair branch could query evaluator metadata. We reran an exact 1,440-row grid on 24 new workflow labels with separate writer, auditor, and policy processes and stores. The broker gives non-oracle policies neither the scenario label nor a truth capability. Active Witness v4 (no truth) records 0% dangerous declarations and 75% closure, matching the full-state oracle; acknowledgement-only fails every silent-no-op and false-closed-read control, while removing terminal settlement fails every post-revocation control. This rules out evaluator and fault-label leakage within the testbed, but single-host separation is not production failure-domain independence.

Dual-auditor quorum and the registered fault bound. The final follow-up replaces a single trusted authority with two independently keyed auditor processes and a separate evaluator. Its 24 new labels cross nine scenarios, two repairability conditions, and six policies (2,592 exact rows). Non-oracle policies again receive neither scenario nor truth. Full quorum requires two valid, distinct evidence groups with one value; it never selects the favorable side of a conflict. When the contract declares the write idempotent and harm-safe across all remaining worlds, it may repair and demand a new quorum. Table 10 reports all policies under the registered at-most-one-faulty-auditor model. Full v5 has zero dangerous declarations in every admissible scenario, detects every false-open/false-closed single-auditor conflict, gains 100 clean safe-closure points, removes 100 clean escalation points, and matches oracle closure. A single-auditor policy fails every false-closed target, and removing settlement fails every post-bundle revocation target. Crucially, when *both* auditors sign the same false closed value, full quorum is dangerous in every control (11.1% over its full grid). This is the observational barrier made executable: quorum is sound only relative to its registered fault bound, not evidence-source naming alone.

Table 10: Capability-isolated dual-auditor confirmation (432 rows per policy; 384 within the registered at-most-one-auditor-fault model). DFs and closure use admissible rows; clean metrics use 48 rows. Best values in each column are bold; ties are all bold.

Policy	DFs ↓	Closure ↑	Clean safe ↑	Clean esc. ↓	Bundles ↓
Passive strict	**0.0**	50.0	0.0	100.0	**1.00**
Acknowledgement only	50.0	50.0	**100.0**	**0.0**	**1.00**
Single auditor A	25.0	62.5	**100.0**	**0.0**	1.39
Quorum, no settlement	12.5	68.8	**100.0**	**0.0**	1.67
Quorum Witness v5	**0.0**	**75.0**	**100.0**	**0.0**	2.28
Full-state oracle	**0.0**	**75.0**	**100.0**	**0.0**	**1.00**

Evidence-domain contract and measured separation. An evidence group is a cryptographic namespace, not by itself a failure domain. We therefore audit separation layer by layer against the frozen service receipt and implementation. Table 11 records both the separated capabilities and shared dependencies. In particular, the two auditors have different processes, files, keys, and group identities but receive events through the same broker-fed update path and share the same host, worker implementation, and controller. The registered one-auditor fault model is exercised by per-auditor corruptions; the common-mode control exercises the consequence of correlated false evidence. Neither establishes the frequency of such faults or independence across production operators.

Table 11: Source-bound evidence-domain audit for the 2,592-row dual-auditor test. "Separated" means exercised as distinct within this testbed, not an assumption of statistical or organizational independence.

Layer	Observed status	Direct evidence	Supported boundary
Process and capability	verified separated	four service PIDs; every policy PID distinct	tests process-local capability and truth isolation
Persistence file	files separated; host shared	four SQLite files under one temporary root	tests file-local state faults, not host or filesystem faults
Signing identity	verified separated	two Ed25519 keys and evidence groups	tests issuer and group binding under retained key secrecy
State-update feed	shared	one broker broadcasts each writer event	does not test a common feed omission or corruption
Implementation and control	shared	one auditor worker and testbed controller	does not test a common code or controller defect
Host, operator, organization	not separated or measured	single-host spawned-process experiment	no production service or organizational independence claim

G.1 Prospective distributed-witness deployment

Before observing outcomes, we froze 30 repeats of 11 fault scenarios and four matched policies. A managed test cluster instantiated the writer and two auditors as exactly three Pods on three distinct nodes from one frozen image. Each process generated its own Ed25519 key and used a Pod-private SQLite store. A pre-outcome receipt hashed the Pod, node, and address identities and required Ready state, the exact image tag, and three-way separation. The runtime then pinned health identities and required one source hash but three distinct Pods, nodes, stores, and public keys. Only after the outcome-blind exact-grid audit passed was the frozen analyzer executed.

Table 12: Prospective cross-node distributed-witness result (30 repeats per scenario; 330 trajectories and 1,320 matched policy evaluations). Declare is the success-declaration rate; DFs is dangerous false success.

Scenario	Policy	Declare (%, descriptive)	DFs (%) ↓
Clean	Dual quorum	100.0	0.0
Auditor A false-closed	Single A	100.0	100.0
Auditor A false-closed	Dual quorum	0.0	0.0
Auditor B false-closed	Single B	100.0	100.0
Auditor B false-closed	Dual quorum	0.0	0.0
Common-mode false-closed	Dual quorum	100.0	100.0

The registered checks all pass: clean quorum declares in 100% of cases; quorum blocks every targeted single-auditor false-closed fault; and it has zero danger over all non-common-mode scenarios. Both auditors signing the same false-closed value produces 100% danger, deliberately retaining the declared impossibility boundary. This validates process-, node-, store-, and key-level separation in one managed test cluster. It is not organizational, operator, cluster, or production independence, and it estimates neither production fault prevalence nor common-mode probability.

G.2 Stateful cross-node action–monitor–recovery study

The distributed-witness experiment above isolates admission logic. We next run the complete post-action loop against a stateful ticket-change service. Each episode executes reserve, cancel, charge, and issue; the monitor can then switch issuer, refund excess payment, roll back, commit, or escalate. Writer acknowledgements, evaluator truth, and signed witness state are separate channels. The acknowledgement comparator is scored on the state immediately after the shared four-action plan, before any GoalSeal probe or recovery, so it does not receive downstream repairs. The controller never observes evaluator truth.

The frozen grid contains 15 conditions with 20 infrastructure repeats each: ten operational conditions combine silent no-op, lost acknowledgement, duplicate effect, or wrong-resource execution with stale, replayed, or wrong-resource witness evidence; four conditions pair an action fault with one persistent false-closed witness; and one condition makes both witnesses report the same false-closed state. A managed test cluster runs the writer and two auditors on three distinct nodes. Every episode pins three different Pod identities, SQLite stores, and Ed25519 keys. Attestations bind workflow, claim, nonce, epoch, resource state, Pod, node, store, and key. These are process and node separations within one organization, cluster, codebase, and service image—not independent authorities.

An initial v3 attempt created no result artifact after a witness request exceeded the original 10-second transport timeout. Before rerunning, v4 froze a 20-second timeout and at most three retries for timeout, connection, or HTTP-5xx failures; stable idempotency keys protect mutating action retries. The complete v4 run passed five of six gates: it produced no dangerous non-common-mode declaration, closed all 200 operational cases, and escalated all 80 persistent single-witness conflicts. However, 40 duplicate-charge episodes escalated with two outstanding charges. We retained this negative result and amended the method before v5: under valid but conflicting evidence, only reversible actions that weakly reduce a lexicographic risk vector in every candidate state and strictly reduce it in at least one are admissible; goal-completing mutations are forbidden. The vector records excess net charge, a cancelled-old/no-issued-new stranded state, and payment exposure for an unfinished replacement. Evidence is invalidated and reacquired after any containment action.

Version 5 prospectively froze the amendment and reran all 300 episodes from zero, reusing no v4 row. Its outcome-blind receipt verifies the exact grid, all seven frozen code/protocol hashes, the fixed image, and three-way Pod/node/store/key separation before analysis.

Table 13: Cross-node stateful repair study (300 episodes). DFs is dangerous false success. "Terminal safe" means at most one net charge; the common-mode row is the registered impossibility control.

Policy / condition	DFs (%) ↓	Resolution (%) ↑	Terminal safe (%) ↑
Acknowledgement only, non-common	78.6	–	–
GoalSeal, operational	0.0	100.0 close	100.0
GoalSeal, persistent one-witness fault	0.0	100.0 escalate	100.0
GoalSeal, common-mode false closure	100.0	0.0 escalate	–

GoalSeal makes 0/280 dangerous declarations outside the common-mode control, compared with 220/280 for acknowledgement. It declares and truly closes 200/200 operational cases, escalates 80/80 persistent single-witness conflicts with no dangerous declaration, and leaves 280/280 non-common-mode states with at most one net charge. It uses 29.3 logical service calls and 32.3 physical HTTP attempts per episode; the completed v5 run records 0 transport retries. The 20/20 common-mode controls remain dangerous because both signed observations are indistinguishable from a closed world in the registered model.

The Wilson intervals and paired exact McNemar calculation describe this fixed grid; the 20 repeats are infrastructure replications, not independently sampled production failures. V5 is a prospective repair validation after observing the v4 containment failure, not an untouched confirmation of the original method. The service and injected faults are researcher-authored, and no airline, payment provider, independent operator, or production traffic is involved.

G.3 Prospective live Redis/HTTP service sanity check

To test whether the closure rule survives contact with real service semantics, we prospectively freeze 60 new resources crossed with eight conditions and four paired policies (1,920 logical episodes). A writer HTTP process mutates a Redis primary; a separately keyed auditor HTTP process reads a distinct Redis replica and signs resource-, version-, epoch-, nonce-, authority-, and evidence-group-bound witnesses. Writer, auditor, primary, and replica have distinct processes, ports, and persistence directories. The monitor has no Redis or evaluator capability. Conditions cover clean close, success-shaped silent no-op and wrong-resource writes, a false-closed writer read, stale/wrong-resource/replayed witnesses, and a signed common-mode false-closed control. The protocol, exact grid, services, runner, analyzer, and witness validators are hashed before the confirmatory run; two analysis executions are byte-identical.

Table 14 reports all registered contrasts. The external witness eliminates observed danger over the seven in-bound conditions and commits every clean case; all stale, wrong-resource, and replayed witnesses fail closed. The signed common-mode control nevertheless causes danger in every case. Thus this experiment narrows the simulator-to-service gap while exposing, not removing, the independence assumption. All components share one host, controller, Redis implementation, and replication stream; this is neither a production prevalence sample nor cross-host or organizational independence.

Table 14: Prospective live Redis/HTTP service sanity check. In-bound DFs exclude only the separately reported signed common-mode control. "Binding esc." is escalation over stale, wrong-resource, and replayed witness cells.

Policy	In-bound DFs ↓	Clean safe ↑	Binding esc. ↑	Common DFs ↓
Acknowledgement only	85.7	100.0	0.0	100.0
Writer-source verification	14.3	100.0	0.0	0.0
External GoalSeal witness	0.0	100.0	100.0	100.0
Full-state oracle	0.0	100.0	100.0	0.0

G.4 External Git-ref closure pilot

Before the first episode, we freeze the protocol, runner, analyzer, five scenarios, three policies, three repetitions, zero-retry rule, and advance criteria. Each episode writes a unique temporary ref to an externally operated repository host, opens a new process for the exact-ref authoritative read, and verifies deletion after the terminal decision. The service URL and organization identity are represented only by a digest in the retained result.

The active witness repairs all silent no-ops and wrong values with one idempotent write, then admits only after a fresh read matches the expected commit. It improves clean committed closure by 50 points and reduces clean escalation by 50 points versus v2 while recording zero observed danger. When the authoritative read is unavailable it escalates in all 3 controls. The repetitions are descriptive service-execution replications, not sampled workflow populations; the injected faults remain researcher-authored. Separate client processes share the same repository service operator and state store, so this is not independent-operator evidence; it does not estimate production prevalence. The full grid contains 45 episodes; the machine-readable artifact retains every policy row and call count.

G.5 Cross-platform composite closure

We prospectively freeze a second real-service study before the first efficacy episode. Its completion contract is conjunctive: a temporary Git ref must equal the expected commit and a separate stateful HTTP workflow must end with the old ticket cancelled, the new ticket issued, and exactly one net

charge. Four policies cross ten scenarios and ten execution repetitions, yielding 400 exact episodes and 4,830 real Git/HTTP calls. Faults independently target either platform with a silent no-op or wrong resource/value, jointly target both domains, or make one required read unavailable. Every Git ref is deleted and independently checked after decision.

Table 15 reports the frozen grid. Dual-domain GoalSeal freshly reads both state stores, permits at most one idempotent repair per mismatching domain, and commits only after both clauses hold. It records 0.0% danger over 60 partial/wrong-state controls and 100.0% clean committed closure. Selective receipts retain only 50.0% clean closure, while Git-only witnessing still records 66.7% danger because service-side failures remain invisible. Both unavailable-witness controls escalate. All seven prospective advance criteria pass and independent analyzer execution is byte-identical.

The repository host and stateful service use distinct protocols, deployments, and state stores under one administrative operator. This is cross-platform failure-domain evidence, not cross-operator or production independence. Faults are researcher-injected and repetitions measure execution stability rather than independently sampled workflow populations.

Table 15: Prospective cross-platform composite-closure experiment. Danger uses six partial/wrong-state scenarios per repetition; clean closure uses two clean scenarios; unavailable escalation uses two witness-outage controls.

Policy	Danger ↓	Clean closure ↑	Outage esc. ↑	Calls ↓	Median s ↓
Acknowledgement only	100.0%	100.0%	0.0%	10.8	7.97
Selective receipt	100.0%	50.0%	100.0%	10.8	7.82
Git witness only	66.7%	100.0%	50.0%	12.5	9.59
GoalSeal dual-domain	0.0%	100.0%	100.0%	14.2	10.17

G.6 External-cloud composite-closure replication

We prospectively replicate the exact 4-policy, 10-scenario, 10-repetition cross-platform design after moving the workflow state store from the managed test cluster to a separately addressed public-cloud VM. The service runs as a dedicated system account, persists to its own SQLite store, binds only to loopback, and is reached through an encrypted SSH tunnel. The scientific fingerprint `314060a4` was committed before the first efficacy episode.

The exact 400-episode run completes in 335 seconds with 4,830 external calls and passes all seven prospective criteria. GoalSeal again records 0.0% danger, 100.0% clean committed closure, and 100.0% final composite closure; Git-only witnessing still records 66.7% danger. Table 16 shows that the registered operating point replicates while external-network latency increases. All temporary Git refs are removed before efficacy analysis.

This is cross-infrastructure, cross-network-location, and cross-state-store evidence, but the VM service was deployed and controlled by the study team. It is therefore not an independent-team reproduction, natural-failure sample, or production deployment.

Table 16: GoalSeal dual-domain operating point across the original managed-cluster execution and the prospectively frozen external-cloud replication.

Environment	Danger ↓	Clean closure ↑	Final closure ↑	Calls ↓	Median s ↓
Managed test cluster	0.0%	100.0%	100.0%	14.2	10.17
External-cloud VM	0.0%	100.0%	100.0%	14.2	7.05

Implementation-diverse paired replication. Before execution, we hash-freeze an exact paired replay of the 2,592-row quorum grid and its Python/Python result comparator. Auditor A retains the original Python, `cryptography`, SQLite, and multiprocessing-queue stack; auditor B is independently implemented in Node.js using native `node:crypto`, a separate canonical serializer, an append-only JSON-lines state log, and JSON-lines stdio RPC. An outcome-blind audit verifies all 24 case-batch receipts, exact order and source hashes, distinct processes, stores, keys, groups, runtimes, implementations, transports, and persistence backends, while requiring the receipts to retain `same_host=true` and `shared_controller=true`. After unlock, every registered advance

check passes and the 18-field scientific outcome projection is row-wise identical; both projections have SHA-256 prefix `739b0065`. Raw and analysis hashes are `9ad3c635` and `88b79e5c`. This removes a shared auditor implementation stack in the replication, but not the shared host, experiment controller, or production-deployment boundary.

Exact quorum fault envelope and common-mode bound. Condition on whether a common-mode false-closure event occurs. With probability c, every evidence domain signs the same false closed value, so any $q \le n$ commits. Outside that event, a false commit with conflict veto requires exactly $f \ge q$ false-closed signatures and $n-f$ abstentions: any remaining valid open-state response would veto the bundle. Independence therefore gives

$$D_{n,q}(p,a,c) = c + (1-c)\sum_{f=q}^{n}\binom{n}{f}p^f a^{n-f}.$$

For a clean closed state, every non-abstaining domain supplies compatible closed evidence, so safe closure is

$$S_{n,q}(a) = \sum_{v=q}^{n}\binom{n}{v}(1-a)^v a^{n-v}.$$

The first expression proves $D_{n,q} \ge c$: adding domains or raising the threshold can shrink only the conditionally independent term. The second makes the associated availability cost explicit.

We enumerate exactly 1,080 frozen design cells over $n \in \{1,2,3,5,7\}$, every $q \le n$, $p \in \{.01,.05,.10\}$, $a \in \{0,.05,.10,.20\}$, and $c \in \{0,.0001,.001,.01,.05\}$. Figure 4 shows the continuous design implication and representative designs. At $p = 5\%$, $a = 10\%$, and $c = 0.1\%$, moving from one source to a 3-source, $q = 2$ conflict-veto quorum changes danger from 5.095% to 0.187% and clean closure from 90.0% to 97.2%, at three evidence calls. By $n = 7, q = 4$, danger is 0.10003%, effectively the 0.1% common-mode floor, while clean closure is 99.7%. These are exact hypothetical design sensitivities, not measured production frequencies.

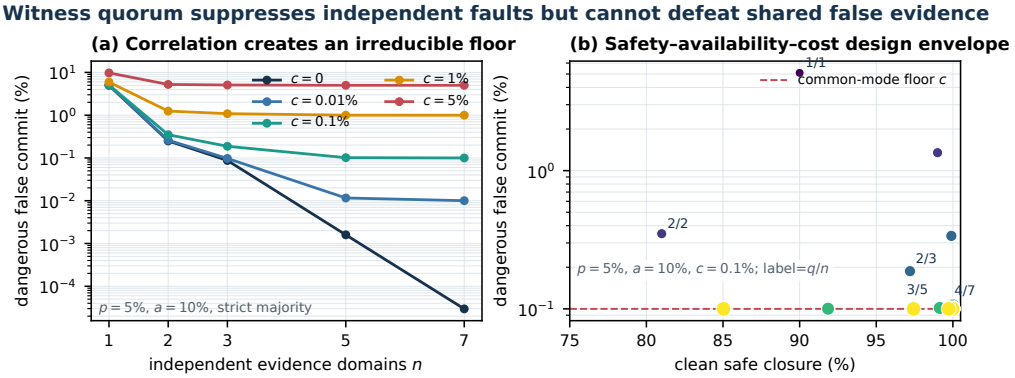


Figure 4: Exact conflict-veto quorum envelope. Left: strict-majority witnesses suppress conditionally independent false evidence but asymptote to the common-mode probability c. Right: all q/n designs for one representative slice; larger bundles trade evidence calls against safety and clean closure. Points are analytic design values, not empirical prevalence estimates.

A pre-hidden recovery-transition diagnostic selects one task per development template and crosses five realized successors with four monitor specifications. The cases are correct recovery, silent no-op, a different open successor, a correlated closed-looking open successor, and a closed-looking successor with an execution-time safety violation. The monitors trust the modeled successor, verify with an optimistic ordinary pair, include the adverse successors without a witness, or add an authoritative witness. The exact 160-row grid contains no stochastic samples. It shows that post-recovery verification detects truthfully observed wrong successors but does not repair an observation model that maps an omitted successor to the same outcomes as a safe closed world. Coverage without a witness preserves safety by abstention; the witness restores nominal coverage. This is a constructed

identification boundary, not evidence that such a witness exists or an estimate of recovery-fault frequency.

G.7 FIVE-WAY RECOVERY-DECISION CONFORMANCE

The runtime's action space is not merely descriptive. We construct 80 exact semantic cases spanning four domains, five decision obligations, and four cost variants. Each obligation has an independently named first-action oracle: continue with a fresh witness when closure remains identifiable, retry when it is the least-cost harm-admissible repair, switch when retry risks duplicating an already committed effect, rollback an unsafe partial effect, or escalate when no informative witness or harm-admissible repair exists. Crossing four policies gives an exact 320-row grid. The executable selector receives only worlds, observation partitions, transition maps, costs, and harm bounds; it does not receive the oracle label.

Table 17: Exact five-way recovery-decision conformance (percent over 80 cases per policy). "Unsafe" means the selected intervention violates the worst-case harm budget; safe handling means closure or fail-closed escalation without realized harm.

Policy	Action match ↑	Unsafe ↓	Dangerous commit ↓	Safe handling ↑
GOALSEAL-CG	100.0	0.0	0.0	100.0
Retry-first	20.0	80.0	0.0	70.0
No max-harm gate	40.0	60.0	0.0	80.0
No witness selection	80.0	0.0	0.0	100.0

GOALSEAL-CG executes all five branches with 100.0% first-action conformance, no unsafe intervention, no dangerous commit, and 100.0% safe terminal handling. Retry-first matches only the retry family and selects a projected-unsafe action in 80.0% of cases. Removing the harm gate makes 60.0% of selections unsafe. Removing witness selection retains 100.0% safe terminal handling but falls to 80.0% action conformance by intervening when observation was the prescribed next step. Thus terminal success alone misses unnecessary mutation, while a zero dangerous-commit score alone misses unsafe recovery attempts. This is a deterministic, controlled semantic conformance result, not a production prevalence estimate, learned-policy comparison, or proof that the supplied world and transition model is complete.

The escalation-packet generator is a read-only sidecar over the selected 864-episode development Agent artifact. It runs only on rows already terminated as `escalate` and may read closure-monitor observations, certificate reasons and source bounds, the effect ledger, recovery count, observed invariant latches, unverified transitions, out-of-support events, and compensation status. Evaluator truth fields are prohibited recursively. The packet contains stable identity, unresolved clauses/reasons, evidence provenance, attempted action and recovery history, residual model-relative risk, and a fail-closed human next action. A v1 representation artifact retained invariant indices only in the risk section; v2 also maps each runtime-visible index to its existing `invariant:i` clause identifier. Both versions are retained, but only v2 is reported. All 183 v2 packets have the five structural sections; 23.5% contain an explicit clause overall, including every GOALSEAL-CG packet. The analysis does not test whether a person understands the packet or makes a better decision.

Industrial dependency versions are frozen in the artifact. Local runtime checks exercise Temporal activity retry, LangGraph checkpointed interrupt/resume, and a complete OpenAI Agents SDK Runner loop with function-tool approval/resume and a tool-output guardrail (Temporal Technologies, 2026; LangChain, 2026; OpenAI, 2026). The latter uses a deterministic test model and makes no provider or production-service call. These checks establish adapter feasibility only; a row is labeled *native* in the main experiment only after the complete episode executes through that runtime with the evaluated model.

Academic provenance is audited at commit granularity. The Verified Tool Calls reimplementation follows its public three-valued verifier, stable idempotency key, and verify-before-retry Algorithm 1; notably, that algorithm accepts a success response without invoking its verifier, which is preserved rather than silently strengthened. The RAC row uses the authors' RecoveryManager at frozen commit `068c327`; its upstream core tests and an explicit-failure retry smoke must pass before evalua-

tion. At artifact freeze, the repository linked by the DART paper resolves to a different transition-guard runtime and does not contain the paper's semantic-recoverability predicates, so we do not label that repository as author code and retain a definition-level reimplementation label.

The hidden split may be opened only after all requested routes have a recorded connectivity disposition, the pilot passes exact-pair and infrastructure-validity checks, prompts and analysis are frozen, and the contract audit is complete or explicitly reported unavailable. The main run uses one deterministically selected instance from each sealed causal template because parameterized surface variants are not independent structural samples. Provider timeouts, rate limits, 5xx responses, malformed tool calls, and route mismatches remain recorded outcomes rather than silently disappearing.

The development contract audit samples 24 items (three per development template). Two annotators independently receive the instruction, initial state, advertised tool effects, allowed state keys, and typed-clause schema. Required-clause agreement is 99.51% symmetric F1 and 99.02% Jaccard, with 95.83% exact agreement by item; invariant clauses agree exactly. One disputed batch-binding predicate is retained by a third adjudicator, so the final match is a resolution of the observed disagreement, not a fresh independent sample. The catalog encodes each safety conjunction as a full-trace invariant; consequently the audit does not validate nonempty Q^- groups. The intervals below are uncertainty summaries, not population-generalization guarantees, because clause observations cluster within the 24 items.

Table 18: Finite human-audit uncertainty. Exact two-sided 95% Clopper–Pearson intervals are Bernoulli sampling-model sensitivity summaries, not population-generalization guarantees: the 24 items are a balanced development convenience sample rather than a probability sample, and clause observations cluster within the 24 items. †Clause-level intervals ignore that clustering and are therefore optimistic. ‡The adjudicated match is reported descriptively without an interval because it reuses the disputed item.

Quantity	Count	Estimate (%)	Exact 95% interval	Unit
Required-clause exact item agreement	23/24	95.8	[78.9, 99.9]	item
Invariant exact item agreement	24/24	100.0	[85.8, 100.0]	item
Annotator A required-clause recall	102/102	100.0	[96.4, 100.0]	clause†
Annotator B required-clause recall	101/102	99.0	[94.7, 100.0]	clause†
Adjudicated required-clause match	102/102	100.0	—	post-adjudication‡

Every executable compiled contract is evaluated on the same realized worlds as its oracle contract. Compiler responses, normalized contracts, prompt hashes, provider errors, clause metrics, generated counterexamples, and downstream outcomes are checkpointed together. Contract calls may use fingerprint-preserving quota batches with the same six-failure circuit break. A complete contract run is ineligible if failed provider calls exceed 5% or form a consecutive block longer than five. The analyzer requires the frozen Cartesian product and pairs compiled/oracle rows by a stable world key.

QUORUMCONTRACT protocol and threshold freeze. The aggregation unit is an exact normalized executable clause, not a free-form semantic judgment. Candidate clauses are canonicalized by predicate type, state key, operator, value, and group structure. A model configuration contributes at most one vote to a canonical clause even if it repeats the clause. Parse failures, schema violations, predicates over unadvertised keys, and non-executable contracts cast no votes and do not shrink the fixed nine-configuration registry. At threshold k, retained clauses require at least k votes; a missing required-clause class makes the aggregate non-executable and therefore fail closed. This rule prevents malformed minority outputs from weakening the acceptance threshold, but it does not make votes statistically independent.

We swept $k = 1, \ldots, 9$ on development only. Thresholds $k \leq 5$ produced zero development dangerous false success, while $k \geq 6$ did not. The strict-majority choice $k = 5$ was more autonomous but failed the zero-dangerous-false-success criterion in four fixed-threshold leave-one-configuration-out conditions. The primary $k = 2$ choice maximized development counterexample accuracy among zero-dangerous-false-success thresholds and retained zero dangerous false success in all nine leave-one-configuration-out analyses. Before hidden opening we froze $k = 2$ as primary and $k \in \{1, 5\}$ as secondary. All three are constructed from the same 144 hidden raw compiler calls; neither a new model call nor a threshold change is permitted after inspecting hidden outcomes.

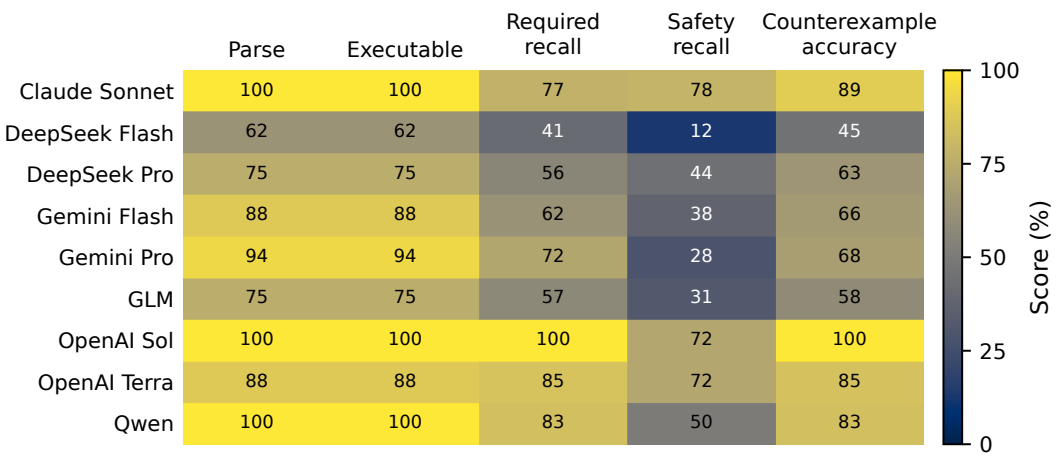


Figure 5: Frozen single-model contract compilation (16 contracts per model). Required-clause recall ranges from 41%–100%, safety recall from 12%–78%, and counterexample accuracy from 45%–100%; all exact-contract rates are zero. Provider success and parseability do not establish semantic goal capture.

H Statistical Analysis

Complete frozen contrasts. Against certification, paired changes were +0.0 pp dangerous false success (95% CI [+0.0, +0.0] pp; Holm $p = 1$), -0.2 pp irreversible harm (95% CI [-0.4, +0.0] pp; Holm $p = 1$), -0.2 pp invariant violation (95% CI [-0.4, +0.0] pp; Holm $p = 1$), and -1.8 pp closure (95% CI [-3.9, +0.2] pp; Holm $p = 0.584$). Escalation changed from 47.7% under certification to 6.8% under counterexample guidance, while mean model calls changed from 13.9 to 11.6. Raw automatic contracts reached 70.3% required-clause recall. Relative to oracle contracts, closure changed by -16.0 pp (95% CI [-20.1, -11.6] pp) and dangerous false success by +1.4 pp (95% CI [+0.3, +2.7] pp); forbidden effects changed by +0.0 pp and invariant violations by -1.2 pp. The frozen $k = 2$ QuorumContract reached 100.0% required-clause recall, changed closure by -6.2 pp (95% CI [-18.8, +0.0] pp), and changed dangerous false success by +0.0 pp (95% CI [+0.0, +0.0] pp), with +0.0 pp forbidden-effect and +0.0 pp invariant-violation changes. In the separate deterministic mechanism track, certified GoalSeal reached 89.8% closure and 0.0% dangerous false success. Its paired closure difference was +4.7 pp versus the independent-evidence verify-before-retry adapter (95% CI [+1.6, +7.8] pp; Holm $p = 0.188$), while dangerous false success changed by -12.5 pp (95% CI [-12.5, -12.5] pp; Holm $p = 0.000244$). Closure changed by +29.7 pp versus logged compensation (95% CI [+26.6, +32.8] pp; Holm $p = 0.000244$). On the prospective 30-task public runtime-certificate track, dangerous declarations under injected faults changed from 36.7% to 0.0% (-36.7 pp; exact sign $p = 0.000977$), while safe closure changed from 23.3% to 16.7% (-6.7 pp; exact sign $p = 0.625$) and escalation from 40.0% to 83.3%. Clean-task safe closure changed from 96.7% to 56.7%; safety, closure, and escalation are interpreted jointly.

Primary estimates are intention-to-treat over all 60 tasks. Fault injection was effective in 513/540 assigned faulty episodes (95.0%); the second table is the mandatory 57-task per-protocol sensitivity.

For each binary outcome, the unit of pairing is a task-level condition within model. We report the observed mean paired risk difference and a 95% percentile interval from 10,000 bootstrap replicates that resample workflow templates and retain all parameterized instances, models, modes, and failures within a selected template. This avoids treating ten instances generated from one structure as independent. The primary two-sided exact sign test reduces all paired differences within each template to one mean effect, discards zero-effect templates, and tests the signs of the remaining template effects; Holm correction is applied across the four registered tests. Exact McNemar tests on discordant condition–model pairs are descriptive. Task-cluster intervals, additional seeds, model strata, and leave-one-template-out analyses are sensitivity checks.

The deterministic mechanism track uses the same template-resampling unit but a separate multiplicity family: certified GoalSeal minus each of two mechanism-equivalent comparators across four outcomes, with Holm correction over eight template-sign tests. The certified-minus-uncertified GoalSeal ablation is corrected across its four outcomes and is never pooled with the Agent main family.

Table 19: Complete frozen mechanism track (128 episodes per policy; 896 total). All policies receive the same oracle contract and realized worlds. Adapters are mechanism-equivalent unless identified as GoalSeal; lower danger, harm, and escalation and higher closure are better. Best values in each column are bold; ties are all bold.

Policy	DFs ↓	Closure ↑	Harm ↓	Esc. ↓	Tool calls ↓
Acknowledgement only	36.7%	50.8%	7.8%	12.5%	**3.3**
Verified Tool Calls adapter	12.5%	60.2%	**7.0%**	27.3%	7.0
Verify-before-retry	12.5%	85.2%	**7.0%**	**2.3%**	8.4
Matched independent verify	12.5%	85.2%	**7.0%**	**2.3%**	8.0
Logged compensation	**0.0%**	60.2%	10.2%	39.8%	8.0
GOALSEAL-CG-uncertified	4.7%	**89.8%**	10.2%	5.5%	9.4
GOALSEAL-CG-certified	**0.0%**	**89.8%**	**7.0%**	10.2%	16.8

Table 20: Complete matched Agent comparison (1,024 eight-model cells × 5 policies = 5,120 episodes). The strict no-commit control is retained here to expose the safety–autonomy tradeoff; it is not a deployable operating point. DFs: dangerous false success.

Method	Origin	DFs ↓	Harm ↓	Inv. ↓	Declare ↑	Esc. ↓	Closure ↑	Calls ↓
Prompted-only	Control	9.4%	6.3%	6.3%	83.1%	5.1%	74.8%	13.5
Verified-call adapter†	VTC-style	5.4%	9.0%	9.0%	88.0%	7.4%	82.7%	11.2
Logged compensation‡	RAC/Saga-style	5.7%	11.2%	11.2%	86.1%	6.2%	80.5%	11.6
GOALSEAL-CG-certified	Ours	0.0%	4.7%	4.7%	0.0%	45.3%	77.5%	15.9
GOALSEAL-CG v3	Ours	0.0%	4.7%	4.7%	82.2%	9.4%	85.0%	15.2

Incomplete, duplicate, or unexpected episode identities fail analysis by default. The analysis program checks the expected Cartesian product and the recorded run fingerprint before computing inferential results.

Safety and coverage are reported jointly rather than treating abstention as success. For each runtime mode we report declaration rate, goal closure, declaration precision, dangerous false success conditional on declaration, escalation, the fraction that attempted an executed mutating action, irreversible harm conditional on such an attempt, model and environment calls, latency, provider-reported tokens and their reporting coverage, provider/protocol failures, maximum-turn termination, and returned-model route mismatches. These are descriptive cost–risk points; monetary cost is not imputed without a frozen route price table. A separate development-only turn-budget sensitivity may diagnose truncation, but additional seeds or budgets are not counted as new structural samples.

I ARTIFACT CHECKLIST

The release is intended to contain the generators, contracts, sealed split hashes, fault schedules, runtimes, simulator, evaluators, contract compiler, QUORUMCONTRACT aggregator, clause registry, baseline provenance, native-adapter manifests, possible-world and recovery-misspecification generators, witness solvers, frozen runners, validators, paired analyses, and tests. It also contains sanitized traces and complete identifier-free projections for the distributed-witness and stateful-repair tracks, with identity aliases, source/audit receipts, and byte-exact replay. Connectivity, execution, resumption, and analysis commands load credentials only from the local environment and never serialize them into results or release archives. Frozen analyzers consume fingerprint-bound raw artifacts and reproduce byte-for-byte the admitted headline outputs. Validity controls preserve the residual boundary: empirical claims are restricted to registered contracts, faults, and evidence channels.

Data, artifacts, and deployment boundary. The benchmark uses synthetic states and public research environments and requires no private production or participant data. The release is designed to include synthetic generators, public contracts, frozen schedules, prompts, analysis code, and non-sensitive sanitized outputs, subject to upstream licenses. For ToolBench-X, it contains our adapter code, revision hashes, protocol, a pseudonymous minimal row-level analyzer input, and identifier-free aggregates rather than upstream tasks or raw trajectories. The strict local gate independently reconstructs this view from the retained raw run; task paths, prompts, answers, model responses, tool traces, and internal route configuration are excluded. Prompt and schema hashes, requested and returned model identifiers, deterministic fault schedules, seeds, provider errors, state traces, and reported usage make every retained private result traceable. Credentials, identifying metadata, and private gold are excluded; anonymized frozen audit responses, role attestations, and their SHA-256

Table 21: Agent operating-point diagnostics (1,152 episodes per mode). Declare is descriptive rather than monotonically preferred. Prec. is closure among success declarations and is undefined when no declaration is made. The short header Fail denotes Prov./protocol fail, including provider and non-provider execution-protocol failures; Max-turn is budget termination. ↑: higher is better; ↓: lower is better. Best values in each comparable column are bold; ties are all bold.

Method	Declare	Prec. ↑	Esc. ↓	Fail ↓	Max-turn ↓	Calls ↓
Prompted-only	**83.0%**	89.0%	**4.7%**	**11.4%**	**1.0%**	**10.8**
GOALSEAL-CG-certified	0.0%	–	47.7%	35.3%	17.0%	13.9
GOALSEAL-CG	70.6%	**100.0%**	6.8%	19.1%	3.8%	11.6

Table 22: Held-out ToolBench-X selective-runtime results by model family. Each model cell contains 60 tasks per environment–mode pair. Faulty dangerous declaration and escalation are lower-is-better; safe closure is higher-is-better. All registered directions are shown.

	Faulty danger (%) ↓			Clean safe closure (%) ↑	
Model	Upstream	v1	v2	v1	v2
OpenAI Terra	33.3	**1.7**	**1.7**	68.3	**83.3**
Gemini Flash	41.7	**1.7**	**1.7**	61.7	**76.7**
GLM	26.7	**1.7**	**1.7**	60.0	**71.7**
All models	33.9	**1.7**	**1.7**	63.3	**77.2**

	Clean escalation (%) ↓		Faulty safe closure (%) ↑	
Model	v1	v2	v1	v2
OpenAI Terra	30.0	**15.0**	18.3	**21.7**
Gemini Flash	38.3	**23.3**	16.7	**20.0**
GLM	38.3	**28.3**	11.7	**18.3**
All models	35.6	**22.2**	15.6	**20.0**

digests are released. Controlled simulation does not authorize consequential deployment: such use additionally requires domain-specific contracts, authoritative evidence channels, access controls, and human oversight.

Research provenance. CRediT roles, funding, and identities will be disclosed in the non-anonymous version. We declare no competing interests or sponsor influence on the benchmark protocol or analysis. The mandatory AI-use disclosure precedes the references.

Table 23: Effective-injection per-protocol sensitivity by model family. Each model cell contains 57 tasks per environment–mode pair. Faulty dangerous declaration and escalation are lower-is-better; safe closure is higher-is-better. All registered directions are shown.

	Faulty danger (%) ↓			Clean safe closure (%) ↑	
Model	Upstream	v1	v2	v1	v2
OpenAI Terra	35.1	**1.8**	**1.8**	70.2	**84.2**
Gemini Flash	43.9	**1.8**	**1.8**	63.2	**77.2**
GLM	28.1	**1.8**	**1.8**	61.4	**71.9**
All models	35.7	**1.8**	**1.8**	64.9	**77.8**

	Clean escalation (%) ↓		Faulty safe closure (%) ↑	
Model	v1	v2	v1	v2
OpenAI Terra	28.1	**14.0**	17.5	**19.3**
Gemini Flash	36.8	**22.8**	15.8	**17.5**
GLM	36.8	**28.1**	10.5	**15.8**
All models	33.9	**21.6**	14.6	**17.5**