BEYOND CORRECTED MEMORY: EXECUTION CONSISTENCY IN MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Shared memory coordinates agents' actions, but correct records do not establish that those actions satisfy task requirements. Memory governance and failure diagnosis regulate or inspect recorded information; they do not by themselves establish whether it is sufficient to judge task duties. We define *execution consistency* through duties governing state use, information handoffs, and final-state agreement, with explicit evidence conditions for judging fulfillment. Our core claim is that identical retained records can correspond to compliant and violating executions under the same task rule. Controlled removal of evidence such as receipt, action dependence, or response validity leaves 82.4% of opposite-label pairs indistinguishable; restoration separates 97.9% of the merged pairs. Natural-log annotations identify the defined violations in actual executions. However, existing logs do not always explicitly represent the execution relationships needed for these judgments. To assess the definition's practical value, we use **CAVERT**, a framework for consistency diagnosis and recovery, to extract supported relationships from logs and apply these criteria. It consistently outperforms contract-prompted LLM and rule-based baselines in diagnosis across all 12 benchmark–executor settings. Under the same gate and executor limits, it also outperforms rule-guided recovery in all four evaluated environments. These findings identify execution evidence that agent-memory and execution interfaces should preserve for reliable judgment.

1 INTRODUCTION

Shared memory lets one agent's observations guide another's actions (Yu et al., 2026b; Margalit et al., 2026). A procurement agent may complete an order after receiving notice that its approval has been revoked. The order succeeds, but the agent has not acted consistently with the updated approval. Judging this behavior requires checking how shared information was used, not just whether the task succeeded or the stored records agree. We ask *what makes execution over shared memory consistent, and what evidence establishes that consistency*.

Memory governance and transactions constrain state, dependencies, and effects (Margalit et al., 2026; Li et al., 2026). Diagnosis and process auditing evaluate available traces (Barke et al., 2026; Cao et al., 2026), while commitment protocols and runtime verification address duties under partial observation (Baldoni et al., 2015; Mahe et al., 2022). The missing link is a criterion for when an execution record contains enough evidence to distinguish duty fulfillment from violation. This requires linking task duties to message receipt, state use, tool effects, and observation coverage.

Our key insight is that *record correctness does not guarantee evidence sufficiency*. A recorded correction need not establish receipt or downstream use; a cancellation request need not establish an effect lasting through the deadline (Figure 1). Thus, **identical retained records can correspond to compliant and violating executions under the same task rule.** Better reasoning over those records cannot recover the missing distinction. Execution consistency therefore treats evidence sufficiency—not record agreement alone—as part of the judgment problem.

We define *execution consistency* as judging task-duty fulfillment from available execution evidence. A judgment must establish the applicable duty, sufficient observation of its execution window, and whether an admissible response fulfilled it; unresolved evidence yields an undetermined outcome rather than presumed compliance or violation. We instantiate this principle across state use, information handoffs, and final-state agreement, covering twelve mechanisms.

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.

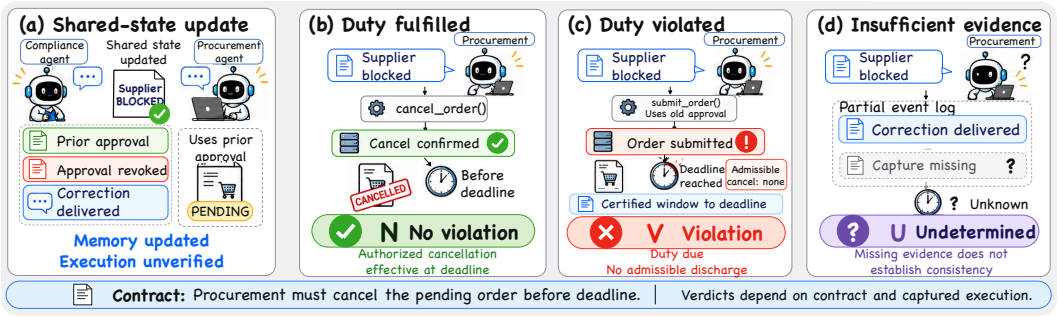

Figure 1: One cancellation duty, three continuations after revocation (a): fulfillment (b, N), non-fulfillment with complete relevant records (c, V), or insufficient evidence (d, U).

To test whether accurate shared-memory contents suffice for judgment, we remove and restore selected execution evidence across 450 controlled pairs, holding each pair's contract, query, and full-execution labels fixed. Removal can merge compliant and violating executions into identical retained inputs; restoration separates nearly all merged pairs. Independent annotations of 360 natural executions confirm that the defined violations occur in practice.

However, these relationships are not always explicit in existing logs. **CAVERT**, a framework for consistency diagnosis and pre-commit intervention, extracts supported relationships, checks source references, and applies the criteria through deterministic rules, retaining unresolved evidence as unknown. Its findings guide recovery. It consistently outperforms contract-prompted LLM and rule-based diagnostic baselines across all 12 benchmark–executor settings, and rule-guided recovery across all four evaluated environments under the same gate and executor limits.

Our contributions are:

1. We **define task-level execution consistency** through duties and evidence of fulfillment, with explicit rules for violation, no violation, and undetermined judgments.
2. We **establish when retained records cannot distinguish compliant from violating runs**, test the required relationships through controlled removal and restoration, and identify the defined violations in natural logs.
3. We **apply these criteria through CAVERT** to diagnose existing logs and guide pre-commit intervention, demonstrating their practical value through diagnosis and recovery experiments.

2 Related Work

State governance. Which state should agents retain and be allowed to use? Distributed consistency constrains operation order, update visibility, and replica convergence (Herlihy & Wing, 1990; Lloyd et al., 2011; Vogels, 2009; Shapiro et al., 2011). Agent-memory research studies retention, access, conflicting records, and visibility (Hu et al., 2026; Margalit et al., 2026; Ren et al., 2026; Volkov et al., 2026; Yu et al., 2026b). TOKI formalizes write-time conflict resolution (Wang, 2026); Khan (2026) verifies a concurrency hierarchy under deterministic generation. MemTX adds belief commit, action gating, and cascading repair (Li et al., 2026). Two runs can share the same approval-register history yet differ in whether the order is canceled. We define the task duties and execution evidence that distinguish such runs, linking shared state to actions, handoffs, and outcomes (Appendix A.5). RQ3 compares CAVERT with MemTX.

Diagnosis and process auditing. Which behavior explains a failure or violates a task requirement? Failure analysis classifies errors and locates responsible agents and steps (Cemri et al., 2025; Zhang et al., 2025; 2026b). AgentRx checks constraints for attribution (Barke et al., 2026); DoVer tests failure hypotheses through interventions (Ma et al., 2026). PAE audits procedural integrity, including agreement between action claims and tool execution (Cao et al., 2026). We focus on shared-memory duties and test whether the retained evidence distinguishes their fulfillment from violation, complementing diagnosis and process audit.

Obligation and behavior verification. Does observed behavior satisfy an explicit rule? Commitment protocols and runtime verification formalize duties, deadlines, and three-valued judgments (Singh, 1999; Mallya & Singh, 2007; Fornara, 2011; Bauer et al., 2011). They also analyze how partial observation affects compliance checking (Baldoni et al., 2015; Mahe et al., 2022). Application-level checks enforce or repair constraints (Balegas et al., 2015; Chang & Geng, 2025); other work checks reasoning, plans, policy compliance, and task actions (Feng et al., 2026a; Wu et al., 2026; Levy et al., 2026; Froger et al., 2026). Building on these semantics, we specify how generated messages, state use, and tool effects establish task-duty fulfillment. Controlled pairs test which relationships must be retained for judgment (Section 3; Appendix A.6).

These lines of work answer different questions. Memory consistency governs what state is retained, exposed, or eligible for use; diagnosis and runtime verification reason about failures or duties from available traces. Execution consistency asks whether the retained trace contains the relationships needed to distinguish fulfillment from violation under a task duty. Our removal–restoration study tests this distinction directly: removing those relationships can make oppositely labeled executions observationally identical to the judge.

3 Execution Contracts and Evidence

Execution consistency is judged relative to a task duty and the evidence available to assess it. It does not require global agreement among agents or memory states, only enough evidence to determine whether the relevant duty was fulfilled, violated, or remains unresolved. The unit of judgment is a task duty within an execution; trace-level consistency aggregates the resulting findings.

The *public contract* P encodes task instructions and declared task or framework policies, fixed before evaluation. For procurement, receipt of an approval revocation activates the duty to cancel the affected order by the final checkpoint. Thus P specifies responsibility, activation, scope, permitted response, and deadline; execution evidence establishes fulfillment.

Three contract families. We group duties by where consistency must hold: at state use, across information handoffs, or among designated final states. *Strict-temporal* contracts govern state use and contradiction resolution at an action or checkpoint. Conflicting writes commit "approved" despite a visible cancellation. Stale dependency use places an order using a replaced approval. Unresolved contradictions leave both states active at a required checkpoint. Contract-permitted exceptions, such as version pinning, are excluded.

Causal-semantic contracts require downstream use to preserve responsibility, evidential support, and scope. A missed handoff ignores a delivered supplier change. An unsupported assumption promises shipment after a failed stock lookup. Unsupported reconstruction presents an unreadable invoice's inferred amount as observed. Source scope loss extends one supplier's evidence to all suppliers. Source-view scope loss treats one page as all orders. Evidence status escalation turns "unverified" into "verified."

Eventual-semantic contracts require agreement among designated final states by a declared endpoint. Unresolved final divergence leaves an order both active and canceled. Competing final answers submit conflicting totals without selecting one. An unresolved final evidence state leaves required checks both complete and pending.

Figure 2 groups twelve candidate mechanisms within these families. Families organize duties; mechanisms identify suspected failures; RQ1's five factors test their evidence requirements (Section 5.1). An execution can involve multiple families. Appendix A.2 gives detailed conditions and exceptions.

Judging these duties requires more than the updated record. Writing a revocation does not establish receipt; receipt does not establish use. A cancellation request does not establish a successful effect lasting through the deadline. No logged cancellation may mean either that none occurred or that its channel was not recorded.

The execution record T preserves these distinctions by linking messages, memory operations, tool calls and results, handoffs, and final actions. It records event order, who could see a state, which actions used it, and which updates replaced it. Coverage records identify captured channels and

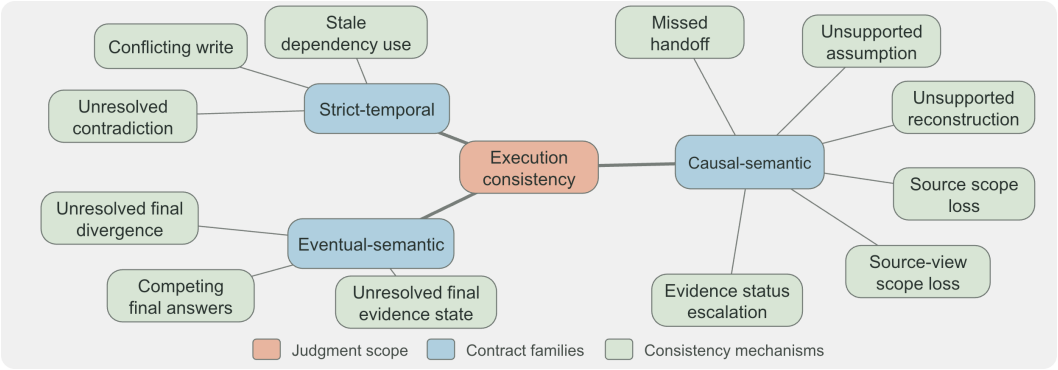

Figure 2: Twelve candidate mechanisms grouped by state use, information handoffs, and final-state agreement. Conditions and exclusions: Table A2.

times. The recording endpoint says when logging stopped, not whether all relevant events were captured. Appendix A.1 gives the formal schema.

3.1 Evidence-Based Consistency Judgments

Return to the procurement agent that orders after receiving the revocation. It must cancel by the checkpoint. An authorized cancellation that succeeds on time and remains effective fulfills this duty: no violation (N). Complete relevant records with no qualifying cancellation establish a violation (V). If the cancellation channel was not fully recorded and fulfillment remains unresolved, neither conclusion is justified: the verdict is undetermined (U). U is an evidential outcome, not a confidence score: the record justifies neither fulfillment nor violation. These judgments concern the cancellation duty, not whether the earlier stale use occurred (Figure 1).

A *finding* F is a suspected problem to check, not a confirmed violation. Given the contract P and execution record T, four questions determine its verdict. The first two identify the problem and duty; the last two assess the evidence and response:

1. **What might be wrong (C)?** Using an old approval is not necessarily a problem. Compare records about the same entity, attribute, and scope (their *semantic key*). Using that approval after a verified replacement became visible to the agent is stale use. Historical references and contract-permitted pinned versions are excluded.
2. **Who must do what, and by when (O)?** A suspected problem does not establish that a response is already due. The contract and activation evidence establish the actor, response, and deadline. An activated duty still binding at its deadline is *active-due*; before then it is pending. Missing, ambiguous, or conflicting policy yields uncertainty, not absence. Established absence of an applicable duty yields N.
3. **Is the relevant process fully observed (W)?** A cancellation may occur without appearing in the log. Check that relevant channels and causal dependencies are recorded, event order is reliable, and state visibility is established. The deadline must be known and recording must extend through it. W is true (a *certified window*) if all six hold, false if any fails, and unknown otherwise.
4. **Was the duty fulfilled, and did the response remain valid (D)?** A cancellation request alone does not establish fulfillment. Check for a response from the responsible or authorized actor, linked to the duty and matching its key and scope. It must occur after activation and by the deadline. Its effect must succeed and remain valid at the deadline or recording endpoint. These conditions define *admissible* fulfillment. Once it is established, the duty is marked *discharged* before judgment (Table A4).

A valid response can establish fulfillment. Establishing non-fulfillment also requires observing where and when a response could occur. Figure 3 therefore checks established fulfillment before requiring a full observation window. The complete rule table is in Appendix A.1.

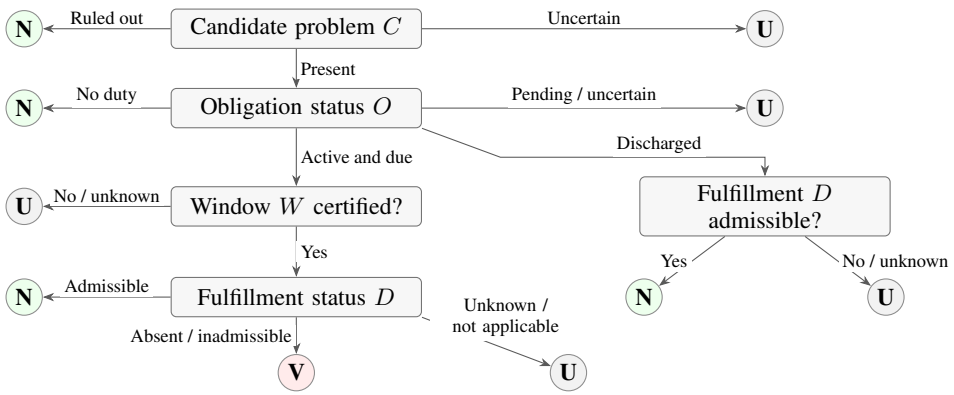

Figure 3: Finding-level decisions. N: no violation; V: violation; U: undetermined. The discharged branch permits N without a window certificate.

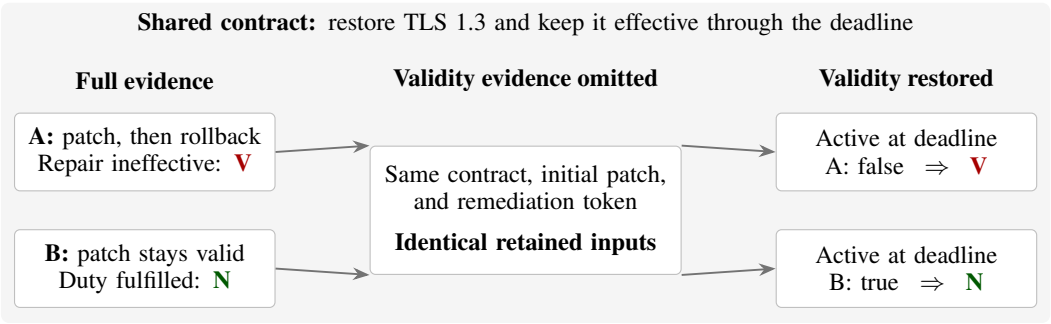

Figure 4: An evaluated pair becomes indistinguishable after removing rollback and validity evidence, then separates after restoration. N/V label the full executions.

In words, V requires an established problem, a binding duty that is due, complete relevant observation, and no response that qualifies as fulfillment:

$$v_F = \mathsf{V} \iff C = \mathsf{present} \wedge O = \mathsf{active_due} \wedge W = \mathsf{T} \wedge D \in \{\mathsf{absent}, \mathsf{inadmissible}\}. \quad (1)$$

At trace level, judgments are aggregated under P: any confirmed violation yields V. With no confirmed violation, any unresolved applicable finding yields U; otherwise the trace receives N. Timely cancellation fulfills the duty without erasing the earlier stale use. Late cancellation establishes recovery, not timely fulfillment.

3.2 Information Needed for a Judgment

The cancellation example requires a response whose effect lasts. AgentWebBench pair `pair_awb_005` isolates the same requirement for repair: restore TLS 1.3 and maintain it through the deadline (Figure 4). Both executions record the same patch and remediation token. A rollback invalidates A's repair (V); B's remains effective (N). Keeping only the contract and initial repair leaves identical inputs. Restoring `active_at_deadline`, the evidence-derived field recording lasting validity, separates them again.

These executions need different answers but look identical to the judge: an *input collision*. Formally, sufficiently observed executions have opposite N/V labels but identical retained inputs under the same contract, finding, and deadline. A deterministic judge restricted to these inputs must answer identically, making at least one error. Better reasoning cannot recover the missing distinction. RQ1 tests which evidence preserves it, independently of CAVERT (Section 5.1; formal criterion in Appendix A.1).

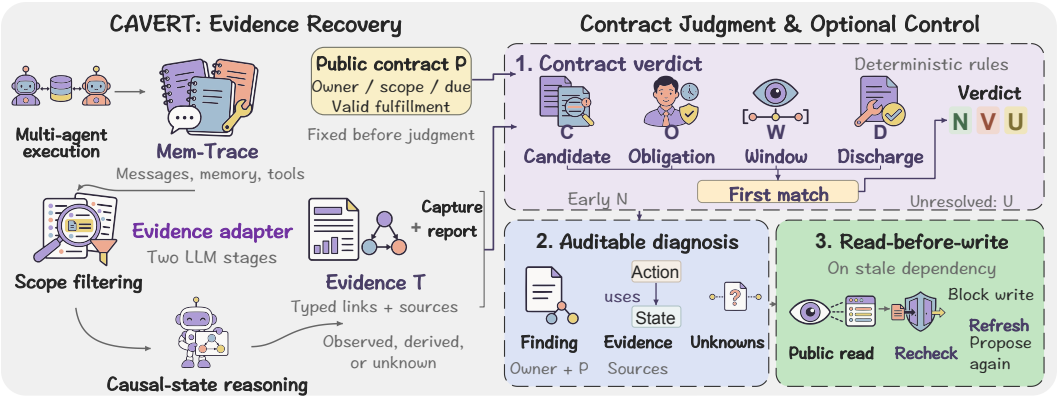

Figure 5: CAVERT recovers execution evidence, applies consistency rules, and optionally gates uncommitted writes. The gate refreshes state before a new action proposal (Section 4.3).

4 Applying the Definition with CAVERT

The definition specifies the evidence required for judgment, but existing logs do not always expose these relationships. CAVERT recovers supported relationships through semantic extraction and source checks, then applies the same deterministic rules. Its findings guide diagnosis, recovery prompts, and pre-commit intervention (Figure 5).

4.1 From Logs to Check States

Given a log and contract P, two LLM stages recover the evidence needed for judgment. Scope filtering selects relevant messages, memory operations, and tool records. Causal-state reasoning links states to recipients, dependent actions, and response effects.

For procurement, records link the revocation's receipt to the later order's use of the old approval (C). The contract supplies the cancellation duty and deadline (O). Capture metadata and adapter checks establish coverage (W); the cancellation's outcome and lasting effect establish fulfillment (D). The adapter retains source-event links and updates the duty's status once fulfillment is established. Unresolved relationships or coverage remain unknown.

LLM interpretation supplies semantic relationships; source checks verify record references (Appendix A.3.1). Structured-trace checks read explicit relationship fields (Appendix A.4.1). Both use the same deterministic verdict rules.

4.2 Applying the Verdict Rules

The validator follows Figure 3 without LLM calls and aggregates verdicts as in Section 3.1. Model and call budgets appear in Appendix B.2.

Each finding reports its verdict, responsible agent, unresolved evidence, and source links (Appendix A.3). To avoid double-counting a duty, CAVERT groups findings only when they share an obligation, compatible keys and scopes, and a verified connecting path. The most specific upstream cause is primary; downstream consequences are secondary.

4.3 From Diagnosis to Recovery

The Core gate checks proposed writes before commitment. In procurement, it blocks an order using superseded approval and identifies the state to refresh through a public read before reproposal. Intervention uses the dependency finding before the deadline, without awaiting a trace-level violation.

Supplementary AppWorld recovery blocks writes to a predefined application until a separate read from that application succeeds. State postconditions and official task success are scored afterwards (Appendix E.4).

Table 1: Information removal merges oppositely labeled executions.

Evidence factor	Removal: merged	Restoration: separated
Consumer receipt	69/80	69/69
Action dependency	65/79	63/65
Actor authorization	68/81	65/68
Historical-version eligibility	57/76	55/57
Repair validity at deadline	69/82	69/69
Total	328/398	321/328

Counts combine three benchmarks. Removal is relative to opposite-label pairs; restoration is relative to merged pairs. Full-view collisions: 4/398. Source breakdowns: Table A7.

The gate assumes uncommitted intercepted writes, side-effect-free rejection, and an appropriate public read. Committed effects require a separate compensation mechanism.

5 Experiments

Our evaluation separates the definition from its realization. RQ1 tests whether the proposed relationships are necessary for judgment; RQ2 whether CAVERT can recover and apply them in existing logs; RQ3 whether the resulting findings support intervention. Formal checks and contract-query results appear in Appendices A.2 and D.

5.1 RQ1: Validity and Necessity of the Definition

Execution relationships carry distinctions needed for judgment. We remove and restore five evidence factors: consumer receipt, action dependency, actor authorization, historical-version eligibility, and repair validity at the deadline. The TLS rollback pair illustrates the last factor: the repair duty stays fixed while evidence of lasting repair is removed and restored (Figure 4).

The study contains 450 pairs from AgentWebBench, MARBLE, and MECoBench (Zhong et al., 2026; Zhu et al., 2025; Liu et al., 2026): 30 per factor per benchmark across 93 task or episode clusters. Each pair shares a contract and query. Blinded human annotation gives 398 opposite-N/V pairs, 38 same-label pairs, and 14 containing U ($\kappa = 0.842$). Removal and restoration keep these full-execution labels fixed (Appendix C).

Among the 398 opposite-N/V pairs, 328 have identical retained inputs after removal (82.4%), compared with 4 (1.0%) before removal. Restoration makes 321 of those 328 input pairs distinguishable (97.9%; Table 1). Task-cluster 95% intervals are [78.1%, 86.2%] and [95.8%, 99.1%], respectively. A judge given identical inputs cannot distinguish the opposite labels; restoring the missing relationships recovers that distinction. We next examine whether the defined violations occur in natural executions.

Natural logs reveal recurring execution-consistency problems. Independent annotation of 360 trajectories (120 each from TraceElephant, AFTraj, and MATM) yields 43 V, 288 N, and 29 U. Among the 43 violating traces, unsupported assumption occurs in 29 (67.4%) and evidence status escalation in 8 (18.6%). Both strengthen downstream claims beyond their supporting evidence, giving concrete targets for causal-semantic checks. Multi-label counting gives 47 trace–mechanism associations across six primary mechanisms (Appendix C.2). This distribution describes the sampled sources, including task-failure-only TraceElephant logs.

Together, these studies show that receipt, use, and valid fulfillment carry distinctions that shared content alone does not preserve. This establishes their necessity, but not whether implicit relationships can be recovered reliably from existing logs. RQ2 tests that gap.

5.2 RQ2: Cross-Source Generalization and Robustness

The evidence requirements support diagnosis from existing logs. We next apply the criteria to 720 executions from MARBLE, CoffeeBench (Sugiura et al., 2026), and MECoBench: four

Table 2: Pooled diagnosis on 720 executions; macro-F1 averages N/V/U equally.

Metric	Generic LLM	Contract LLM	Rules	**CAVERT**
Accuracy (%)	64.2	80.6	78.9	**87.1**
Macro-F1 (%)	43.5	74.8	73.8	**83.5**

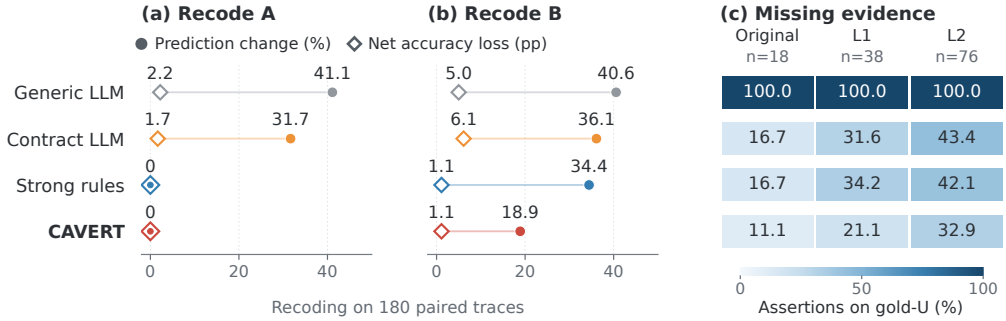

Figure 6: Diagnosis under log transformations (180 traces). (a,b) Lines join prediction-change rates and net accuracy losses, not confidence bounds. (c) N/V answer rates on reference-U cases; denominators are shown. Lower is better. Full results: Table A12.

execution models and 60 runs per benchmark–model setting. Diagnostic configurations stay fixed across sources. Generic and Contract LLM use DeepSeek-V4-Flash; CAVERT uses Qwen3.5-Flash filtering, DeepSeek extraction, and deterministic judgment (Appendix B).

CAVERT reaches 87.1% accuracy versus 80.6% for Contract LLM and 78.9% for Rules (Table 2). It leads in accuracy and macro-F1 across all 12 settings (Table A12), supporting the criteria's use across the tested sources.

Because call structures differ, we also compare under a common $0.0120 per-execution spending limit. CAVERT exceeds Contract LLM + review by 3.47 pp accuracy and 0.049 macro-F1; both paired 95% intervals exclude zero. Mean API costs are $0.0017 versus $0.0031 (Appendix D.4). The advantage persists when the baseline can review and revise within the same spending limit.

Format and missing evidence. We next test verdict stability under recoding and judgment under reduced evidence. A separate panel gives 180 traces their original view plus four transformations (900 inputs). Recode A renames identifiers and reorders JSON keys; Recode B uses a line-based format with causal pointers. Both preserve relationships and labels. Missing L1 removes delivery confirmations and handshake receipts; L2 also removes pre-deadline invalidations and intermediate action confirmations. Unlike RQ1, reference labels and completeness flags are reassessed from the remaining evidence (Appendix D.2).

CAVERT leads all three baselines in accuracy and macro-F1 across all five views (Table A12). All its predictions remain unchanged under Recode A; 146/180 remain unchanged under Recode B, versus 118/180 for Rules (Figure 6). The 18 correct-to-wrong and 16 wrong-to-correct transitions yield a 1.1 pp accuracy decline. Individual verdicts remain sensitive to format despite the small net change.

Under Missing L2, CAVERT reaches 66.1% accuracy versus 57.2% for Rules. It nevertheless assigns N to 25/76 reference-U and 18/52 reference-V cases: the former are unsupported no-violation judgments; the latter miss violations supported by retained evidence (Appendix D.3). Diagnosis alone does not establish practical value; RQ3 asks whether these findings improve action under fixed intervention authority.

5.3 RQ3: Practical Benefits and Intervention Cost

Consistency findings guide recovery. After evaluating diagnosis, we test its use in pre-commit intervention. In procurement, a stale-approval finding directs the agent to refresh the approval state

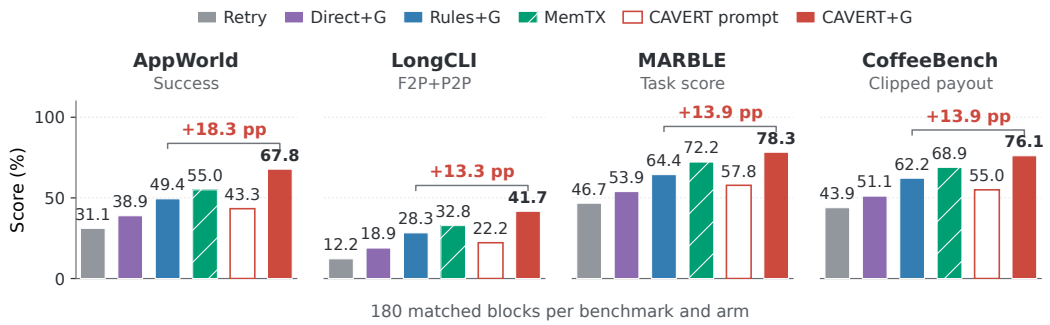

Figure 7: Recovery on 180 matched blocks per benchmark. G: shared gate. Gains are CAVERT+G minus Rules+G before rounding (Table A19).

before revising its proposed order. Comparing CAVERT with Rules under the same gate tests the value of this guidance.

We use 720 matched fault blocks, 180 each from AppWorld (Trivedi et al., 2024), LongCLI (Feng et al., 2026b), MARBLE, and CoffeeBench (Figure 7). Within each block, arms share initial states, action budgets, output ceilings, and timeouts. Aggregate scores equally weight benchmark-specific normalized endpoints (Appendix E.1).

With the same gate, CAVERT improves on Rules+G in all four environments. AppWorld success rises from 89/180 to 122/180 (+18.3 pp), and LongCLI repair without regressions from 51/180 to 75/180 (+13.3 pp). Consistency findings therefore improve recovery with intervention authority held fixed.

We also compare complete systems with their own intervention mechanisms. CAVERT-Gate exceeds MemTX by 8.75 pp in mean normalized score (95% CI [6.39,11.11]) on the same blocks. All four benchmark intervals are positive (Table A20).

The benefit of gating depends on diagnostic guidance. Holding continuation length fixed, we test three diagnostic methods with prompting and with gating. Switching to gating adds 17.7 pp with CAVERT and 8.6 pp with Rules. The difference between these gains is 9.1 pp (95% CI [4.8,13.4], $p < 0.001$; Table A22). Thus, the same gate is more effective when guided by consistency findings.

Intervention should also preserve normal-task performance. On 480 blocks without injected faults, the CAVERT-minus-Native score difference is -0.7 pp (95% CI $[-2.1, +0.7]$). The lower bound exceeds the predeclared -2.5 pp loss margin, supporting aggregate non-inferiority (Table A23).

Recovery incurs computational overhead. On a separate long-horizon cohort, CAVERT+G adds 24.6% tokens and 5.0 seconds per task over Native. Supplementary AppWorld/LongCLI billing averages \$0.042 versus Native's \$0.030 per task, including cache discounts (Table A28).

RQ1 identifies evidence needed to distinguish fulfillment from violation; RQ2 and RQ3 show how it supports diagnosis and corrective action. Together, the results connect evidence sufficiency to judgment and intervention in multi-agent systems. This progression matters because the same evidence that makes execution behavior judgeable also makes failures actionable before commitment.

6 Conclusion

Correct memory is not enough for consistent execution. Identical records can conceal different task-duty outcomes, and no downstream judge can recover distinctions the record failed to preserve. We formalize this gap as execution consistency and identify the evidence needed to judge it. CAVERT shows that these requirements support diagnosis and corrective action. The broader design question is therefore not only whether memory is correct, but whether it preserves enough evidence to judge execution. Reliable multi-agent memory should preserve not only what agents know, but how shared information was received, used, and acted upon.

REPRODUCIBILITY STATEMENT

Appendix A specifies the semantics and evidence schema. Supplementary material provides an implementation snapshot, extraction prompts, an executable example, and 432 constructed executions with reference labels. Paired intervals preserve the stated clustering units. Appendices B and E specify configurations.

ETHICS STATEMENT

This work studies execution consistency, diagnosis, and recovery in multi-agent systems using research benchmarks, controlled executions, and model-generated or benchmark-provided traces. We do not collect new human-subject, personal, or sensitive data. Our experiments evaluate task-level execution behavior and system interventions rather than decisions about individuals or demographic groups. Recovery mechanisms operate within the evaluated benchmark environments and are not designed for deployment in consequential real-world settings. We use benchmark data and models for research evaluation and report the evidence requirements, intervention scope, and experimental protocols needed to interpret our results. We follow the ICLR Code of Ethics and aim to support transparent and reproducible evaluation of multi-agent systems.

AI USE STATEMENT

Generative-AI tools assisted code, writing, and figure preparation. Reference labels for the information-pair and diagnostic studies were independently human annotated. The authors are responsible for experimental records, references, and final claims.

REFERENCES

Peter Bailis, Alan Fekete, Michael J. Franklin, Ali Ghodsi, Joseph M. Hellerstein, and Ion Stoica. Coordination avoidance in database systems. *Proceedings of the VLDB Endowment*, 8(3):185–196, 2014. URL https://www.vldb.org/pvldb/vol8/p185-bailis.pdf.

Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, and Munindar P. Singh. Composing and verifying commitment-based multiagent protocols. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 10–17, 2015. URL https://www.ijcai.org/Proceedings/15/Papers/009.pdf.

Valter Balegas, Sérgio Duarte, Carla Ferreira, Rodrigo Rodrigues, Nuno Preguiça, Mahsa Najafzadeh, and Marc Shapiro. Putting consistency back into eventual consistency. In *Proceedings of the Tenth European Conference on Computer Systems*, 2015. URL https://www.dpss.inesc-id.pt/~rodrigo/indigo_eurosys15.pdf.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026. URL https://arxiv.org/abs/2602.02475.

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. URL https://arxiv.org/abs/2506.07982.

Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):1–64, 2011. doi: 10.1145/2000799.2000800.

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. URL https://arxiv.org/abs/2603.03116.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia,

Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4082.

Edward Y. Chang and Longling Geng. SagaLLM: Context management, validation, and transaction guarantees for multi-agent LLM planning. *Proceedings of the VLDB Endowment*, 18(12):4874–4886, 2025. doi: 10.14778/3750601.3750611.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. *arXiv preprint arXiv:2604.22708*, 2026. URL https://arxiv.org/abs/2604.22708.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. VeriCoT: Neuro-symbolic chain-of-thought validation via logical consistency checks. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=zHuV3Vatov.

Yukang Feng, Jianwen Sun, Zelai Yang, Jiaxin Ai, Chuanhao Li, Zizhen Li, Fanrui Zhang, Kang He, Rui Ma, Jifan Lin, Jie Sun, Yang Xiao, Sizhuo Zhou, Wenxiao Wu, Yiming Liu, Pengfei Liu, Yu Qiao, Shenglin Zhang, and Kaipeng Zhang. LongCLI-Bench: A preliminary benchmark and study for long-horizon agentic programming in command-line interfaces. *arXiv preprint arXiv:2602.14337*, 2026b. URL https://arxiv.org/abs/2602.14337.

Nicoletta Fornara. Specifying and monitoring obligations in open multiagent systems using semantic web technology. In *Semantic Agent Systems*, pp. 25–45. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-18308-9_2.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c26a67e0470774df98c12480ec5d2d7b-Abstract-Conference.html.

Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463–492, 1990. doi: 10.1145/78969.78972.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fd1eff9dd295df50a41f2521942fa31d-Abstract-Conference.html.

Sajjad Khan. Verified detection and prevention of concurrency anomalies in multi-agent large language model systems. *arXiv preprint arXiv:2606.17182*, 2026. URL https://arxiv.org/abs/2606.17182.

To Eun Kim, Xuhong He, Dishank Jain, Ambuj Agrawal, Negar Arabzadeh, and Fernando Diaz. Multi-agent transactive memory. *arXiv preprint arXiv:2606.19911*, 2026. URL https://arxiv.org/abs/2606.19911.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MuCDzH0ctf.

Xiaoyang Li, Yiqi Wang, Haohui Lu, Zhi Chen, Mo Li, Pingan Song, Mingkai Zheng, and Taotao Cai. MemTX: Transactional belief commit for stateful agent memory. *arXiv preprint arXiv:2607.23929*, 2026. URL https://arxiv.org/abs/2607.23929.

Qingyun Liu, Jiwen Zhang, Jingyi Hu, Siyuan Wang, and Zhongyu Wei. MECoBench: A systematic study of multimodal agent collaboration in embodied environments. *arXiv preprint arXiv:2606.31966*, 2026. URL https://arxiv.org/abs/2606.31966.

Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 2011. URL https://www.cs.princeton.edu/~mfreed/docs/cops-sosp11.pdf.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mrEK16Jy6h.

Erwan Mahe, Boutheina Bannour, Christophe Gaston, Arnault Lapitre, and Pascale Le Gall. Dealing with observability in interaction-based offline runtime verification of distributed systems. *arXiv preprint arXiv:2212.09324*, 2022. URL https://arxiv.org/abs/2212.09324.

Ashok U. Mallya and Munindar P. Singh. An algebra for commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, 2007. doi: 10.1007/s10458-006-7232-1.

Yanki Margalit, Nurit Cohen-Inger, Erni Avram, Ran Taig, and Oded Margalit. Governed shared memory for multi-agent LLM systems. *arXiv preprint arXiv:2606.24535*, 2026. URL https://arxiv.org/abs/2606.24535.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. GateMem: Benchmarking memory governance in multi-principal shared-memory agents. *arXiv preprint arXiv:2606.18829*, 2026. URL https://arxiv.org/abs/2606.18829.

Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems*, pp. 386–400. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-24550-3_29.

Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999. doi: 10.1023/a:1008319631231.

Issa Sugiura, Daichi Hattori, Kazuo Araragi, Keita Ogawa, Shota Onose, Taro Makino, Teppei Usuki, and Takashi Ishida. CoffeeBench: Benchmarking long-horizon LLM agents in heterogeneous multi-agent economies. *arXiv preprint arXiv:2606.16613*, 2026. URL https://arxiv.org/abs/2606.16613.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850.

Werner Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009. doi: 10.1145/1435417.1435432.

Sergey Volkov, Yang Li, and Ye Luo. StateFuse: Deterministic conflict-preserving memory for multi-agent systems. *arXiv preprint arXiv:2607.05844*, 2026. URL https://arxiv.org/abs/2607.05844.

Ziming Wang. TOKI: A bitemporal operator algebra for contradiction resolution in LLM-Agent persistent memory. *arXiv preprint arXiv:2606.06240*, 2026. URL https://arxiv.org/abs/2606.06240.

Feiyu Wu, Xu Zheng, Yue Qu, Zhuocheng Wang, Zicheng Feng, and Hui Li. Grounding generative planners in verifiable logic: A hybrid architecture for trustworthy embodied AI. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wb05ver1k8.

Shuhan Xue, Zixin Ding, Yichen Shen, Yinjie Wang, Zhenfei Yin, Yingcheng Wu, Yuxin Chen, Mengdi Wang, and Ling Yang. PAST-Bench: Benchmarking the foundations of recursive self-improvement in personal agents. *arXiv preprint arXiv:2608.04003*, 2026. URL https://arxiv.org/abs/2608.04003.

Tao Yu, Hao Wang, Changyu Li, Shenghua Chai, Minghui Zhang, Zhongtian Luo, Yuxuan Zhou, Haopeng Jin, Zhaolu Kang, Jiabing Yang, YiFan Zhang, Xinming Wang, Hongzhu Yi, Zheqi He, Jing-Shu Zheng, Xi Yang, Yan Huang, and Liang Wang. Beyond the all-in-one agent: Benchmarking role-specialized multi-agent collaboration in enterprise workflows. *arXiv preprint arXiv:2605.08761*, 2026a. URL https://arxiv.org/abs/2605.08761.

Zhongming Yu, Naicheng Yu, Hejia Zhang, Wentao Ni, Mingrui Yin, Jiaying Yang, Yujie Zhao, and Jishen Zhao. Multi-agent memory from a computer architecture perspective: Visions and challenges ahead. *arXiv preprint arXiv:2603.10062*, 2026b. URL https://arxiv.org/abs/2603.10062.

Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. AgentForesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026a.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=l05DseqvuD.

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, pp. 76583–76599, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Shanshan Zhong, Kate Shen, and Chenyan Xiong. AgentWebBench: Benchmarking multi-agent coordination in agentic web. *arXiv preprint arXiv:2604.10938*, 2026. URL https://arxiv.org/abs/2604.10938.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, 2025. doi: 10.18653/v1/2025.acl-long.421.

APPENDIX CONTENTS

A Semantics and Implementation 15

A.1 Evidence and decision rules 15

A.2 Mechanism registry 16

A.3 Trace schema and adapters 17

A.4 Obligations and effective discharge 22

A.5 Relation to distributed consistency 23

A.6 Relation to agent diagnosis and verification 23

B Shared Evaluation Protocol 23

B.1 Annotation, metrics, and uncertainty 23

B.2 Diagnostic and execution configurations 24

B.3 Semantic extraction diagnostic probe 25

C Definition Evaluation Details 25

C.1 Paired design and information removal 25

C.2 Mechanism occurrence in natural logs 26

D Diagnosis and Robustness Details 27

D.1 Contract-query diagnosis (E2) 27

D.2 Cross-model and multiview diagnosis (E3) 28

D.3 Prediction stability and evidence sufficiency 30

D.4 Diagnosis under a fixed spending limit 31

E Execution Protocol and Stratification 32

E.1 Matched blocks and executor configurations 32

E.2 Recovery controls and outcome definitions 34

E.3 Normal-task impact and budget sensitivity 34

E.4 Supplementary cohorts, outcomes, and costs 36

A Semantics and Implementation

A.1 Evidence and Decision Rules

A.1.1 Execution Record and Information Loss

The execution record is

$$T = \langle E, \mathrm{Prec}, \mathrm{Causal}, \mathrm{Vis}, \mathrm{Sup}, P, \mathrm{Capture}, H \rangle. \tag{2}$$

Here E denotes events, Prec reliable precedence, and Causal typed dependencies. Visibility $\mathrm{Vis}(i, a, j)$ records whether agent a could observe event i before event j. Supersession Sup and contract P determine which states and effects remain eligible for use. $\mathrm{Capture}$ records which channels and segments were captured. The observation horizon H is the point where recording ends. Records are compared using the semantic key $K(e) = \langle \mathrm{entity}, \mathrm{attribute}, \mathrm{scope} \rangle$.

Retained-record indistinguishability

Fix a public contract P, a target finding F, and an evaluation horizon h. The recorded horizon H does not guarantee coverage through h. Let $q(T)$ retain a specified subset of execution information. Let $\ell_P(F, T, h)$ be the evidence-supported reference judgment. Suppose sufficient evidence establishes opposite labels $\ell_0, \ell_1 \in \{\mathsf{N}, \mathsf{V}\}$ under the same (P, F, h), but $q(T_0) = q(T_1)$. Any deterministic judge g restricted to that input then satisfies

$$\sum_{j=0}^{1} \mathbf{1}[g(P, F, h, q(T_j)) \neq \ell_j] \geq 1. \tag{3}$$

Identical inputs require identical predictions, matching at most one label. The criterion tests whether a representation preserves verdict-relevant distinctions.

A.1.2 Evidence Domain

Every primitive queried by the semantics is either observed in the raw execution, reproducibly derived by a versioned adapter rule, or unknown. Observed and derived values can evaluate to true or false; unknown evidence evaluates to U. The window certificate uses strong Kleene conjunction to combine these values. Any false requirement makes the certificate false. If none is false but one is unknown, the certificate is unknown. It is true only when all requirements are true. The finding verdict then follows the ordered finite-trace monitor below. An uncertified window leaves missing discharge records insufficient to establish non-fulfillment. Earlier decisive rules can still establish N without evaluating the window.

For the truth ordering $\mathsf{F} < \mathsf{U} < \mathsf{T}$,

$$x \wedge y = \min(x, y), \qquad \neg\mathsf{T} = \mathsf{F}, \quad \neg\mathsf{F} = \mathsf{T}, \quad \neg\mathsf{U} = \mathsf{U}.$$

A.1.3 Visibility, Reachability, and Frontier

VisAt is the relation Vis in Eq. (2), evaluated for an agent before an event. True visibility requires one of three grounds: a produced i, an active access-control rule includes a, or explicit delivery makes i visible before j. An explicit exclusion yields F only with complete delivery capture through j and no intervening delivery to a. Otherwise visibility is unknown. An empty or missing access-control list is never interpreted as globally visible. Typed causal reachability is computed over data_dependency, control_dependency, handoff_delivery, evidence_support, supersession, and resolution edges. Temporal adjacency or repeated speech by one agent does not create a causal edge. The typed edges in Causal record execution relationships; Sup records which earlier states have retired. They are not interchangeable: a dependency or delivery edge need not retire a state.

A *frontier* contains states still eligible for use: recorded replacements, retractions, or resolutions have not retired them. For trigger j and semantic key k, the global pre-trigger frontier is

$$\mathrm{Frontier}_{pre}(j, k) = \{i \mid i \prec j,\ K(i) \sim k,\ \nexists r : i \prec r \prec j \wedge \mathrm{Retires}(r, i)\}. \tag{4}$$

Retirement requires an explicit supersession, retraction, version replacement, or effective resolution. When retirement is unknown, frontier membership is unknown. The agent-visible frontier uses only

Table A1: Ordered stage states and their finding verdicts. Decisive earlier stages short-circuit later checks.

Candidate	Obligation	Window	Discharge	Final
Absent	Not evaluated	Not evaluated	Not applicable	N
Unknown	Not evaluated	Not evaluated	Unknown	U
Present	Absent	Any	Not applicable	N
Present	Pending or uncertain	Any	Any	U
Present	Discharged	Any	Admissible	N
Present	Discharged	Any	Not admissible or unknown	U
Present	Active due	Closed	Admissible	N
Present	Active due	Closed	Inadmissible or absent	**V**
Present	Active due	Closed	Unknown or not applicable	U
Present	Active due	Open or unknown	Any	U

N: no violation for this mechanism; V: violation; U: undetermined. "Any" means the value cannot change this row's verdict; "not evaluated" denotes a short-circuited stage, not an unknown truth value.

the agent's observation history. A retirement changes this frontier only when the agent can see it:

$$\begin{aligned}\mathrm{Frontier}^{a}_{pre}(j,k) = \{i \mid i \prec j,\ K(i) \sim k,\ \mathrm{VisAt}(i,a,j) = \mathsf{T},\\ \nexists r:\ i \prec r \prec j \wedge \mathrm{Retires}(r,i) \wedge \mathrm{VisAt}(r,a,j) = \mathsf{T}\}.\end{aligned}$$

An unseen retirement does not silently remove a state from a's view. Unknown visibility or retirement makes any affected membership unknown; downstream predicates may not treat this as established absence. Checkpoint mechanisms use the frontier named by the public contract. A consumer-specific stale-use finding instead requires the replacement to be visible to that consumer before use.

A.1.4 Stage Truth Table and Trace Aggregation

At trace level, one confirmed finding is sufficient for VIOLATION. If no finding is a violation but any applicable mechanism remains UNDETERMINED, the trace is UNDETERMINED. The trace is NOVIOLATION only when every applicable mechanism has a determined NOVIOLATION verdict, including verdicts established by short-circuiting. Trace aggregation therefore preserves any unknown mechanism that could change the verdict.

A trace-level N concerns the consistency mechanisms applicable under P. The recovery gate enforces read-before-write sequencing for uncommitted writes (Section 4.3); action authorization remains a separate policy requirement.

A.2 Mechanism Registry

Table A2 specifies the evidence requirements and exclusions for the twelve mechanisms introduced in Section 3. A required handoff may bind its recipient when it uses the information, while explicitly permitted alternatives need not be reconciled.

Table A2: Required evidence and exclusion conditions for the twelve consistency mechanisms.

Mechanism	Required structural evidence	Exclusion or boundary
Strict-temporal: consistency at a dependency or reconciliation checkpoint		
Conflicting write	Visible active frontier write; overlapping key/scope; contradictory committed downstream write	Labelled alternatives or an explicit resolution are not conflicting writes
Stale dependency use	Verified retirement of old state; downstream dependency on it; replacement visible before use	Explicit version pin, authorized rollback, or historical qualification
Unresolved contradiction	Two contradictory active frontier states at a declared reconciliation checkpoint	Resolution before the checkpoint prevents this checkpoint candidate
Causal-semantic: preservation of responsibility, support, and evidence scope		
Missed handoff	Typed handoff with payload, target, delivery condition, and later target dependency action	Preserve, update, qualify, or resolve before the dependency action
Unsupported assumption	Failed, partial, unknown, or unverified evidence followed by a stronger dependent commitment	Independent evidence sufficient for the stronger claim
Unsupported reconstruction	Corrupt or unrecoverable source followed by reconstruction presented as observed or verified	Independent support or an explicitly hypothetical reconstruction
Source scope loss	Source scope is known; downstream scope exceeds it; no support covers the expansion	Scope-preserving or explicitly narrowed claim
Source-view scope loss	Paged, ranked, filtered, windowed, or top-k view generalized to the global source	Global completeness is separately established
Evidence status escalation	Failed, partial, unknown, or unverified evidence upgraded to gathered, complete, or verified	New successful evidence or an auditable verification step
Eventual-semantic: consistency of the effective terminal state		
Unresolved final divergence	Incompatible task-relevant terminal frontier states remain active at horizon	Specific upstream mechanism is primary when it explains the same obligation
Competing final answers	Conflicting typed final answers with no canonical selection at horizon	Policy-authorized labelled alternatives can make the obligation non-binding
Unresolved final evidence state	Terminal evidence availability, completeness, or verification states disagree and affect output	New evidence, qualification, or resolution before horizon

The 12×4 executable suite tests four cases per mechanism: a violation satisfying Eq. (1), a normal execution without the candidate problem, an authorized and timely discharge, and insufficient evidence requiring UNDETERMINED. Discharge preserves an event-time candidate but can prevent a later checkpoint or terminal candidate. These remain distinct findings even when they concern the same key; grouping follows Section 4.2.

A.3 TRACE SCHEMA AND ADAPTERS

The normalized trace is serialized as external-adapted-trace-1.0. Each record contains a stable trace identifier, source metadata, a capture manifest, capability evidence, and an ordered event array. Diagnosis operates on events and their relations, without preselecting a message pair.

For each finding, CAVERT returns

$$y_F = \langle m, \pi, C, O, W, D, v_F, e_r, e_t, \rho \rangle, \tag{5}$$

where m is the mechanism, π the responsible principal, e_r the root event, e_t the trigger or due-point event, and ρ a typed dependency path. Evidence references and abstention reasons accompany the observed path from state break to affected action. Eq. (1) specifies decisions conditional on these fields; extraction quality is evaluated empirically.

Adapters return both the normalized trace and a capability report. Visibility, semantic keys, obligation force, and causal paths retain their observed or derived support; missing values remain unknown.

Table A3: Public trace fields and their role in evidence construction.

Object	Required fields	Purpose
Source	source_id, framework, framework_version, raw_record_id	Locks provenance and permits source- and framework-stratified evaluation
Capture manifest	adapter_version, channel states, horizon and truncation states, raw-log reference	Establishes which negative observations are justified
Event	event_id, sequence, event_type, actor, content, fields	Represents messages, memory operations, tools, retrieval, browser views, handoffs, and finalization
Field evidence	status, value, provenance, confidence, reason	Separates observed, derived, and unknown inputs and makes derivations auditable
State identity	state_group_id, state_dimension, state_label, semantic key/scope	Supports comparisons between heterogeneous sub-states without inventing a global task key
Graph evidence	typed parents, visibility activation, supersession, resolution, and support links	Supports causal localization, retirement, and discharge-effect checks

Policy comes from explicit task requirements, and observation coverage is checked separately for each finding. Derived fields store the adapter rule and source event identifiers needed for replay.

A.3.1 Extraction Instructions and Structured Outputs

The open-log implementation separates evidence selection, candidate localization, and finding-conditioned review within the selection and reasoning workflow of Section 4.1. These operations can involve multiple page-level calls; model assignments and per-call limits are specified in Appendix B.2. The following excerpts are taken from the implemented prompts.

Evidence selection

`STATE_ATOM_PROMPT` instructs: "This is exhaustive evidence indexing, not violation judgment" and "Do not discard weak, failed, tentative, partial, or contradictory states: record their epistemic status." It retains claims, source evidence, shared writes, corrections, handoffs, dependent actions, and final states. Each returned atom has the form `[event_id, atom_type, semantic_label, epistemic_status, evidence_quote]`. Event IDs must come from the supplied page; quotes must be literal source spans, or indexed span IDs resolved by the runtime. An `overflow` flag requests page splitting when the atom limit is exceeded. Actor identity and source text are restored from the original events.

Candidate localization

`CROSS_TRACE_PAIRING_PROMPT` compares the indexed events and instructs: "Do not decide obligation, deadline, discharge, or final violation." It lists the twelve registered mechanisms and requires localized root, trigger, and path event IDs. Ordinary task failure or an incorrect answer without a supported execution relationship is insufficient. Its JSON response contains `recovered_candidates`; each candidate names a mechanism, a typed key or cited-span identity, and the supporting events. A candidate is a proposal for checking, not a final verdict.

Finding-conditioned review

`FORMAL_V4_TARGETED_PROMPT` instructs: "Review the entire supplied page separately for obligation, window, and discharge." A public instruction or rule must bind the duty; a normally observed final action must establish its deadline. The prompt further specifies: "A search retry, acknowledgement, critique, or correction marker without later uptake is not discharge." For each finding, it returns one `finding_reviews` entry with `obligation`, `window`, and `discharge` assessments. Each assessment contains a status (`supported`, `not_supported_on_page`, or `uncertain`), support event IDs, an initially empty citation list, and a short rationale. The request repeats the public task, completion metadata, and finding anchors; these anchors do not count as evidence that intervening pages were scanned.

The inventory and pairing prompts are defined in `experiments/formal_v4_exhaustive_candidate_recovery.py`; the targeted prompt and output normalization are in `experiments/formal_v5/semantic_evidence.py`. Their separation prevents the extraction interface from substituting an LLM's final label for the deterministic judgment.

A.3.2 From Responses to Checked States

Normalization and source checks. After JSON parsing, `_normalize_response` normalizes candidate identities and path boundaries and applies candidate and predicate JSON schemas. For model-selected event IDs, `_hydrate_citations` copies source text into `exact_quote`; the model need not transcribe it. The open-log adapter then checks that the mechanism is registered, root/trigger/support IDs exist, required references are cited, quotes match source records, and path events have valid order. Predicate evidence must match an accepted candidate. Invalid or conflicting items cannot establish the affected state; unresolved evidence is retained rather than treated as absence.

Certifying negative evidence

`not_supported_on_page` is a page-local search result, not a false predicate by itself. To establish absent discharge, the targeted verifier records all covered event IDs, page counts, completed reviews, and a deadline event. The adapter checks complete coverage of the supplied record, one negative review per page, and an existing deadline; a positive discharge review prevents a negative certificate. An incomplete or uncertain search cannot establish absent discharge. Scanning all supplied text does not establish that an unrecorded execution channel was captured; capture adequacy and task-specific window evidence remain separate requirements.

Field conversion. After these checks, the open-log adapter constructs the following states. "Supported" below means accepted evidence after normalization and audit, not the unverified JSON assertion alone.

Field	Conversion used for judgment
C	An accepted, confirmed localized candidate yields present; rejected or unresolved proposals yield unknown. Established absence or refutation is distinguished from missing evidence.
O	Supported binding and due obligation yields active_due; established absence yields absent; unresolved binding yields uncertain.
W	Accepted finding-window evidence supplies the window certificate and its supporting event references; open or unresolved windows do not become closed merely because extraction finished.
D	Supported admissible fulfillment yields admissible; a certified negative search yields absent; otherwise discharge remains unknown. An active-due duty with admissible fulfillment is marked discharged before judgment.

`SemanticOpenLogV4Validator` in `memory_consistency/v4/semantic_open_log.py` performs the audit and conversion. `final_status` in `memory_consistency/v4/models.py` applies Table A1; it makes no LLM call. Source checks verify reference integrity and structural constraints. They do not prove that every proposed semantic relationship, duty interpretation, or window assessment is correct; extraction quality remains an empirical question (Appendix B.3).

A.3.3 EXECUTABLE INPUT–OUTPUT EXAMPLE

This constructed example illustrates the complete candidate and predicate response formats and their deterministic processing. The responses are specified fixtures, not sampled model outputs or additional evaluation observations. The trace is fully observed through a normal final checkpoint, with public completion metadata `final_event_id=e4` and capture metadata permitting negative-evidence search.

Example input: four recorded events

ID	Actor / event	Recorded content, in order
e1	User / instruction	Use only verified supplier eligibility for the final approval.
e2	Planner / memory write	Supplier eligibility is assumed; verification is unavailable.
e3	Procurement / approval	Approved supplier eligibility based on the assumed status.
e4	Procurement / final checkpoint	Final approval checkpoint reached; the approval remains active.

Candidate response. The proposed `unsupported_assumption` links the assumed state in e2 to its consumption in e3. The cited-span identity uses text present in e2. The complete illustrative response is:

Candidate response **JSON fixture**

```
{"recovered_candidates": [{
  "mechanism": "unsupported_assumption",
  "semantic_key": null,
  "semantic_identity": {
    "mode": "cited_span", "label": "Supplier eligibility",
    "event_ids": ["e2"]
  },
  "root_event_ids": ["e2"], "trigger_event_ids": ["e3"],
  "causal_path_event_ids": ["e2", "e3"], "citations": []
}]}
```

Normalization retains the selected events and fills their source citations. For example, the e2 citation becomes `event_id="e2"` with `exact_quote="Supplier eligibility is assumed; verification is unavailable."` It does not add a new semantic assertion.

Predicate response. The finding-conditioned review selects e1 for the obligation and e4 for the window; the negative discharge assessment covers this example's single page. Its complete illustrative response is:

Predicate response **JSON fixture**

```
{"finding_reviews": [{
  "mechanism": "unsupported_assumption",
  "root_event_ids": ["e2"], "trigger_event_ids": ["e3"],
  "predicate_reviews": {
    "obligation": {
      "status": "supported", "support_event_ids": ["e1"],
      "citations": [],
      "rationale": "The public instruction requires verified
     eligibility."
    },
    "window": {
      "status": "supported", "support_event_ids": ["e4"],
      "citations": [],
      "rationale": "Normal completion closes the approval duty."
    },
    "discharge": {
      "status": "not_supported_on_page", "support_event_ids": [],
      "citations": [],
      "rationale": "All events reviewed; no propagated repair occurs."
    }
  }
}]}
```

Runtime-generated certificate and verdict. The verifier copies source citations for the selected IDs and converts the supported obligation and window to true predicate evidence. It generates the following discharge-search certificate; coverage counts are runtime bookkeeping, not model-provided verdicts:

Discharge-search certificate **Runtime-generated**

```
{"mechanism": "unsupported_assumption",
 "root_event_ids": ["e2"], "trigger_event_ids": ["e3"],
 "deadline_event_id": "e4",
 "covered_event_ids": ["e1", "e2", "e3", "e4"],
 "page_count": 1, "completed_page_count": 1,
 "search_conclusion": "no_admissible_discharge",
 "page_reviews": [{"page_index": 0,
   "status": "no_admissible_discharge_on_page",
   "rationale": "All events reviewed; no propagated repair occurs."}]}
```

Checked states and resulting verdicts

Replaying these fixtures through normalization, `TargetedFindingVerifier`, and `SemanticOpenLogV4Validator` gives:

Input condition	Checked states / consequence	Verdict
Original fixtures	$C = \mathsf{present}$, $O = \mathsf{active_due}$, $W = \mathsf{T}$, $D = \mathsf{absent}$	**V**
Only `completed_page_count` changed to zero	The negative certificate is invalid; D becomes unknown.	**U**
Required candidate quote absent from the source	C becomes unknown.	**U**

These checks demonstrate the conversion and rejection behavior; they do not measure an LLM's ability to produce the illustrated responses.

Table A4: Public obligations and their permitted effects. A: policy binding. B: observable conditions for discharge.

A. Binding policy

Claim or policy state	Default binding interpretation
Hard constraint, verified fact, or evidence state	must_preserve; exploratory context does not relax provenance, source scope, supersession, or evidence-status invariants
Recommendation	should_consider only when policy requires justification, resolution, or no unresolved deviation
Hypothesis or brainstorm	Optional only when exploration is explicit and no invariant is contradicted
Labelled alternatives	Non-binding only when the relaxation names the mechanism/scope and declares that no canonical resolution is required
Missing, ambiguous, or conflicting policy	Obligation is uncertain; no hidden domain threshold is substituted

B. Permitted discharge

Mode	Required observable effect
preserve	Downstream state retains the obligated payload and evidence strength
update	Stronger or newer evidence replaces the old state while retaining provenance
resolve	Conflicting branches are explicitly retired and one effective state is established
qualify	Normative force or scope is narrowed to what the evidence supports
authorized_defer	Policy permits delay and the event names owner, condition, and a strictly later deadline

All modes require an authorized actor, the same key and scope, a typed causal path, and an effect after activation and no later than the deadline. The effect must remain active at the evaluation frontier.

A.4 Obligations and Effective Discharge

Contracts are fixed before judgment from explicit task instructions and declared policies. Controlled pairs share a case-level specification; external logs use public task metadata or user-instruction messages. Each obligation records its identifier, key and scope, responsible principals, authorized resolvers, activation, deadline, discharge modes, and scoped exceptions. Execution evidence separately establishes agent visibility, activation, and fulfillment. Due status follows these conditions rather than a coarse policy profile.

An obligation is absent when public policy establishes that no relevant responsibility exists. It is pending before its deadline. It is uncertain when a binding fact is unknown. The adapter assigns active_due once grounding, target responsibility, activation, and due status are established. Confirming an admissible discharge changes that state to discharged. "Not due" therefore means pending, not absent. The adapter updates O before verdict evaluation. An established due obligation with admissible D is supplied as $O = \mathsf{discharged}$. The final status function reads C, O, W, D without rewriting O. Independently supplied states may instead contain $O = \mathsf{active_due}, D = \mathsf{admissible}$ (Table A1). This combination requires a closed W. The two rows thus represent different inputs, not a choice of rules for the same normalized evidence. For a present candidate and unknown W, the discharged input yields N; the active-due input yields U.

Later events can revoke a discharge; the permitted effect must still hold at the deadline or horizon (Table A4B).

Early N and persistence evidence

"Early" denotes rule order, not a judgment made before future effects are known. For a present candidate, $O = \mathsf{discharged}$ and $D = \mathsf{admissible}$ establish N without requiring the full W certificate. Admissibility still requires evidence of authorization, matching key and scope, causal connection, timing, and continued effect through the evaluation point. A successful repair receipt alone does not establish persistence if subsequent capture is incomplete.
For example, an agent receives a replacement at event 1, uses the old state at event 2, and repairs it at event 3 before deadline 9. A lasting authorized repair with complete relevant subsequent records establishes discharge. Explicit audience records can establish local visibility while global coverage, and hence W, remains unknown. If persistence is unresolved, D is unknown and this branch yields U. If the repair is retired by event 9, no alternative response exists, and W is closed, the verdict is V.

A.4.1 Evidence Paths and Evaluation Roles

The formal-v4 `final_status` function applies Table A1 to C, O, W, D. E2/E3 recover these states through two LLM stages and score the complete log-to-verdict path (Appendix B.2).

Structured-trace checks read explicit fields instead. Audiences or completed typed delivery establish visibility; supersession and effect-retirement links track response persistence. Persistence requires a known horizon, no truncation, and complete order, causal, memory, tool, and message records. Global visibility coverage is checked separately by W. Missing required links or persistence evidence leave D unknown.

A.5 Relation to Distributed Consistency

Linearizability concerns an object's sequential specification (Herlihy & Wing, 1990). Two executions can share the same approval-register history yet differ in whether procurement cancels an order after receiving a revocation. A specification covering order actions can enforce that duty. Our judgment concerns its fulfillment, beyond the register history alone.

COPS tracks dependencies before exposing updates (Lloyd et al., 2011). Our causal-semantic contracts additionally check meaning: a correctly ordered handoff can still turn "filtered records" into "all records." Likewise, eventual convergence (Vogels, 2009) differs from resolving designated task states by a specified deadline. These distinctions concern the specification and evidence, rather than a stronger consistency model over the same object.

Application invariants and semantic checks already support coordination, enforcement, and repair (Bailis et al., 2014; Balegas et al., 2015; Chang & Geng, 2025). Commitment protocols and runtime verification supply obligation lifecycles and three-valued judgments (Singh, 1999; Mallya & Singh, 2007; Fornara, 2011; Bauer et al., 2011). We specify the evidence linking generated handoffs and tool outcomes to these duties, and test its role through controlled pairs. U covers unresolved execution evidence or policy binding.

A.6 Relation to Agent Diagnosis and Verification

DoVer tests failure hypotheses through interventions, while AgenTracer attributes failures to agents and steps (Ma et al., 2026; Zhang et al., 2026b). VIRF checks plans against safety constraints; Veri-CoT checks formalized reasoning; ST-WebAgentBench evaluates policy-compliant task completion (Wu et al., 2026; Feng et al., 2026a; Levy et al., 2026). CAVERT judges task-obligation fulfillment from visibility, dependencies, response effects, and coverage.

B Shared Evaluation Protocol

B.1 Annotation, Metrics, and Uncertainty

E1 varies information under a fixed public contract. E2 scores queries registered before method predictions. E3 preserves original labels for lossless views and judges missing-evidence views from what remains. E1 uses two independent annotators with agreement $\kappa = 0.842$ and third-party

Table A5: E2–E3 diagnostic and execution configurations.

Component	Model / engine	Inference configuration
B3	deepseek-ai/DeepSeek-V4-Flash-2026-07	$T = 0$, top-$p = 1$, 2,048 output tokens; thinking disabled
B4	deepseek-ai/DeepSeek-V4-Flash-2026-07	$T = 0$, top-$p = 1$, 4,096 output tokens; thinking disabled
Rules	`PublicContractRules`	Deterministic public-contract checks; unknown required evidence propagates to U
AgentRx	AgentRx-Magentic-One-v4.0 + adapter	Adapter: agentrx-magentic-v4-loss-aware-1.0
CAVERT	S1: Qwen3.5-Flash; S2: DeepSeek-V4-Flash	$T = 0$, top-$p = 1$, 4,096 output tokens per model call in each stage; scope filtering then causal-state inference
Execution agent	deepseek-ai/DeepSeek-V4-Flash-2026-07	$T = 0.2$, top-$p = 0.95$, 4,096 output tokens; ReAct loop

B3/B4 use direct and contract prompts. Diagnostic configurations share public inputs and use the method-specific output ceilings above; CAVERT's ceiling applies per call. These are limits rather than measured consumption. Core E4 executor settings appear in Appendix E.

adjudication of disagreements. The development sets contain 30 pairs for E1 and 90 traces for E2 and do not enter formal denominators.

Metric definitions

Three-way macro-F1 averages the N, V, and U classwise F1 scores equally; zero-denominator classes receive zero.

Decision coverage (Cov.) is the fraction of all units answered N or V, including reference-U units. It measures definite-answer frequency rather than log completeness.

Binary violation F1 excludes reference U: a predicted U on V is a false negative, while a predicted U on N is not a false positive.

Task-cluster resampling keeps each task's arms, queries, models, seeds, and views together. E1 uses 10,000 percentile-bootstrap replicates; all-zero or all-one cells yield degenerate empirical intervals.

B.2 Diagnostic and Execution Configurations

B3/B4 both disable thinking, with different output ceilings (Table A5). Rules (also labeled Strong rules) uses `PublicContractRules` to evaluate public-contract evidence deterministically and propagate unknown required evidence to U. AgentRx uses a contract resolver because failure attribution differs from N/V/U judgment. CAVERT's two LLM stages supply scope and causal-state reasoning; the final status function maps the recovered check states to N/V/U deterministically.

Stage interfaces. Given the log and contract, Qwen3.5-Flash selects relevant messages, memory operations, and tool records. DeepSeek-V4-Flash links this evidence to recipients, dependent actions, and effects. The adapter constructs T and finding-level C, O, W, D states, retaining supporting record references and leaving unresolved fields unknown. The validator then applies Table A1 and aggregates findings without further LLM calls. The 4,096-token output ceiling applies to each extraction call, not to the full two-stage process.

The execution-table labels DeepSeek, Gemini, Qwen, and Llama denote DeepSeek-V4-Flash, Gemini-2.5-Flash, Qwen3.8-27B-FP8, and Llama-3.3-70B-Instruct-FP8, respectively. These are execution models; Qwen3.5-Flash is used in extraction. The four-model diagnosis and supplementary recovery panels are distinct from Core E4's two executor configurations, specified in Appendix E.

Table A6: System configurations and a reference-semantic-input probe ($N = 432$ traces; 144 task clusters).

Configuration	Evidence Source	Adjudication Rules	3-Way Acc	Error Count	Violation Prec / Rec / F1
1. Structural Baseline	Heuristic (No LLM)	Heuristic Rules	58.3%	180 errors	45.8% / 64.7% / 0.537 (FP=117)
2. End-to-End CAVERT	2-stage LLM	Deterministic Rules	**83.3%**	**72 errors**	**91.7% / 64.7% / 0.759 (FP=9)**
3. Oracle + Rules	Reference semantics	Deterministic Rules	**97.9%**	**9 errors**	**100.0% / 100.0% / 1.000 (FP=0)**

The structural baseline changes both extraction and adjudication, making it a full-system comparison. CAVERT reduces FP from 117 to 9, while recall remains 64.7% (TP=99, FN=54). The non-deployable Oracle probe has 9 errors versus 72 end-to-end errors: a net reduction of 63 (87.5%). Its remaining errors are Gold N predicted as U; violation FP/FN are zero on this set. With rows and columns ordered V/N/U, the Oracle matrix is $((153, 0, 0), (0, 171, 9), (0, 0, 99))$, giving macro-F1 0.977.

B.3 Semantic Extraction Diagnostic Probe

The Oracle probe tests evidence quality under fixed verdict rules on a separate calibration set of 432 traces from 144 task clusters (Table A6). It replaces CAVERT's two-stage LLM extraction with reference semantic inputs. The structural baseline changes both extraction and adjudication.

Reference semantic inputs raise accuracy from 83.3% to 97.9% and reduce errors from 72 to 9 (87.5%); the remaining errors are N cases predicted U. RQ1 identifies relationships that preserve verdict distinctions; this probe shows the value of supplying them accurately. Together, they motivate recording these relationships during memory use, message delivery, and tool execution, so future checkers can read them directly. CAVERT provides the complementary log-based route for existing systems.

C Definition Evaluation Details

C.1 Paired Design and Information Removal

The three sources each contribute 150 pairs across five factors. The official query and public contract are the same within a pair. Consumer receipt and version eligibility matter only where the contract makes them relevant; an unreceived update is not by itself proof of a violation. A repair must remain admissible at the deadline, rather than merely have appeared earlier. Full, reduced, and selectively restored representations are canonicalized while preserving semantically relevant identity and ordering. Here, *full* denotes the normalized representation before ablation, not the complete execution record. Equality is tested on this representation; reference labels assess duty fulfillment in the underlying executions under the shared contract.

Reference judgments assess duty fulfillment using the pre-removal evidence and remain fixed across views. Each ablation removes the selected relationship; restoration adds back its verdict-relevant information from the full view. For example, `active_at_deadline` in Figure 4 summarizes whether the repair remains valid at the deadline after accounting for later events.

The 398 opposite-label pairs contain 4 full-view collisions, 328 reduced-view collisions, and 321 restored separations. Every reported factor cell satisfies $S \leq C_{\text{minus}} - C_{\text{full}}$. This bound assumes reduction is a deterministic function of the full representation and restoration introduces no information outside that full view.

Table A7 reports all benchmark–factor cells, each containing 30 pairs. The 38 same-label pairs and 14 pairs containing U remain in the collected sample but are excluded from collision denominators.

Table A7: Full sample accounting and representation collisions by information factor.

Information factor	D/A	C_{full}/D	C_{minus}/D	S/C_{minus}
AgentWebBench				
Consumer receipt	27/30	0/27	23/27	23/23
Action dependency	26/30	0/26	21/26	21/21
Actor authorization	28/30	1/28	24/28	23/24
Historical-version eligibility	25/30	0/25	19/25	18/19
Repair validity at deadline	27/30	0/27	22/27	22/22
Subtotal	133/150	1/133	109/133	107/109
MARBLE				
Consumer receipt	27/30	0/27	24/27	24/24
Action dependency	27/30	1/27	23/27	22/23
Actor authorization	26/30	1/26	22/26	21/22
Historical-version eligibility	26/30	0/26	20/26	19/20
Repair validity at deadline	28/30	0/28	24/28	24/24
Subtotal	134/150	2/134	113/134	110/113
MECoBench				
Consumer receipt	26/30	0/26	22/26	22/22
Action dependency	26/30	1/26	21/26	20/21
Actor authorization	27/30	0/27	22/27	21/22
Historical-version eligibility	25/30	0/25	18/25	18/18
Repair validity at deadline	27/30	0/27	23/27	23/23
Subtotal	131/150	1/131	106/131	104/106
Total	398/450	4/398	328/398	321/328

A: collected pairs; D: opposite N/V labels; S: reduced collisions separated by restoring information. The 38 same-label and 14 U-containing pairs remain in sample accounting but not in D.

Table A8: Reference verdicts in the 360-trace natural-log panel.

Source	Sampling unit and stratum	Traces	V	N	U
TraceElephant	Released MAS task-failure traces	120	38	82	0
AFTraj	Safe / diagnosed natural-failure traces	120	2	105	13
MATM	One population run per sampled task	120	3	101	16
Total		360	43	288	29

V/N/U are consistency reference labels, not task-success labels or detector predictions. Each source has its own sampling frame.

C.2 Mechanism Occurrence in Natural Logs

The natural-log annotation release, frozen on September 2, 2026, contains 360 traces: 120 each from TraceElephant (Chen et al., 2026), AFTraj (Zhang et al., 2026a), and MATM (Kim et al., 2026). Two annotators independently assess the twelve mechanisms without seeing CAVERT predictions; a third adjudicates disagreements. The separate 30-trace calibration set is excluded.

TraceElephant contributes 40 Captain-Agent and 80 Magentic-One task-failure traces. AFTraj contributes 90 source-labeled safe and 30 diagnosed natural-failure traces, sampled within domains. MATM contributes one randomly selected population run per sampled task: 40 ALFWorld and 80 WebArena tasks. Injected AFTraj traces, MATM expert demonstrations, and single-agent SWE-Agent traces are excluded. Source labels and task outcomes are hidden during annotation. The source strata define the descriptive sample; its proportions characterize these selected logs rather than deployment-wide prevalence.

Table A9 counts a trace once for a mechanism when that mechanism's final reference verdict is V. Secondary manifestations are excluded from these primary-mechanism counts. Four violating traces have two primary mechanisms, yielding 47 trace–mechanism associations across 43 violating traces. Percentages therefore use 43 as the denominator and need not sum to 100%.

Table A9: Primary violation mechanisms in natural logs.

Mechanism	TE	AF	MT	Total	% of 43 V
Strict-temporal					
Conflicting write	0	0	0	0	0.0
Stale dependency use	0	0	0	0	0.0
Unresolved contradiction	0	0	0	0	0.0
Causal-semantic					
Missed handoff	0	0	0	0	0.0
Unsupported assumption	24	2	3	**29**	**67.4**
Unsupported reconstruction	1	0	0	1	2.3
Source scope loss	3	0	0	3	7.0
Source-view scope loss	4	0	0	4	9.3
Evidence status escalation	8	0	0	8	18.6
Eventual-semantic					
Unresolved final divergence	0	0	0	0	0.0
Competing final answers	2	0	0	2	4.7
Unresolved final evidence state	0	0	0	0	0.0
Trace–mechanism associations	42	2	3	47	

TE: TraceElephant; AF: AFTraj; MT: MATM. Counts are distinct traces per primary mechanism with a V reference verdict. A trace can contribute to multiple rows. Secondary labels are excluded; zero means no primary V label in this sample.

Table A10: E2: Comparison over 930 contract queries across three benchmarks.

Method	MARBLE	CoffeeBench	MECoBench	Pooled	Δ [95% CI]
Always N	0.205	0.204	0.207	0.206	-0.545 $[-0.578, -0.512]$
Generic LLM	0.465	0.463	0.470	0.466	-0.285 $[-0.318, -0.252]$
AgentRx + resolver	0.591	0.591	0.591	0.591	-0.160 $[-0.191, -0.130]$
Strong rules	0.632	0.631	0.622	0.629	-0.122 $[-0.150, -0.094]$
Contract LLM	0.663	0.660	0.657	0.660	-0.091 $[-0.117, -0.064]$
CAVERT	0.753	0.747	0.753	0.751	Reference

All score columns are three-way macro-F1; benchmark query counts are 348, 312, and 270. Pooled uses the combined matrix. Differences are method minus CAVERT pooled macro-F1, with paired 95% intervals. Accuracy, violation F1, and coverage are in the stratified appendix table. AgentRx includes a contract resolver.

Unsupported assumption is the most frequent primary mechanism: 24 TraceElephant, 2 AFTraj, and 3 MATM traces. Evidence status escalation is next, with 8 TraceElephant traces. Both concern downstream claims that exceed their supporting evidence. Unresolved final evidence state also appears as a secondary manifestation in 25 traces; its zero primary count reflects this distinction. The other zero entries denote no primary V label in this sample. Together, the annotations show that the definition identifies recurring execution problems in natural logs while distinguishing them from task failure alone.

D Diagnosis and Robustness Details

D.1 Contract-Query Diagnosis (E2)

The supplementary E2 panel evaluates diagnosis at the contract-query level: 930 preregistered queries on 360 executions, 120 each from MARBLE, CoffeeBench, and MECoBench. Query counts are 348, 312, and 270; pooled labels are 425 V, 415 N, and 90 U. Our AgentRx adaptation (Barke et al., 2026) uses a contract resolver to map failure attribution to N/V/U judgments.

Table A11: E2: Diagnostic performance by benchmark.

Method	Macro-F1	Binary V F1	Acc. (%)	Cov. (%)
MARBLE ($N = 348$)				
Always N	0.205	0.000	44.5	100.0
Generic LLM	0.465	0.749	66.4	100.0
AgentRx + resolver	0.591	0.630	64.9	90.5
Strong rules	0.632	0.714	70.1	87.1
Contract LLM	0.663	0.731	72.7	91.1
CAVERT	0.753	0.783	78.2	91.1
CoffeeBench ($N = 312$)				
Always N	0.204	0.000	44.2	100.0
Generic LLM	0.463	0.750	66.0	100.0
AgentRx + resolver	0.591	0.629	64.7	90.4
Strong rules	0.631	0.709	69.6	86.9
Contract LLM	0.660	0.727	72.1	90.7
CAVERT	0.747	0.770	77.2	90.4
MECoBench ($N = 270$)				
Always N	0.207	0.000	45.2	100.0
Generic LLM	0.470	0.752	67.4	100.0
AgentRx + resolver	0.591	0.620	65.2	90.7
Strong rules	0.622	0.715	70.4	88.1
Contract LLM	0.657	0.730	73.0	91.5
CAVERT	0.753	0.776	78.1	91.5

Macro-F1 averages N/V/U equally. Binary V F1 excludes reference-U cases; a predicted U on V is a miss. Cov. is the percentage of all queries answered N or V.

E2: CAVERT confusion matrices

Rows are reference labels and columns are predictions, both ordered V,N,U. CAVERT's three benchmark matrices are

$$M_{\mathrm{MB}} = \begin{pmatrix} 108 & 46 & 5 \\ 9 & 142 & 4 \\ 0 & 12 & 22 \end{pmatrix}, \quad M_{\mathrm{CB}} = \begin{pmatrix} 94 & 43 & 5 \\ 8 & 126 & 4 \\ 0 & 11 & 21 \end{pmatrix}, \quad M_{\mathrm{ME}} = \begin{pmatrix} 83 & 37 & 4 \\ 7 & 112 & 3 \\ 0 & 8 & 16 \end{pmatrix}.$$

CAVERT leads in macro-F1; the generic judge has higher violation recall. Table A11 reports per-benchmark metrics. Table A10 pools confusion matrices and reports method-minus-CAVERT differences with intervals from matched query predictions.

D.2 Cross-Model and Multiview Diagnosis (E3)

The multiview evaluation contains 180 traces, four methods, and five views (3,600 predictions). Accuracy, macro-F1, coverage, and prediction transitions use the same reference labels and predictions, matched by benchmark, executor, and trace within each view. AgentRx is evaluated in the query-level setting; the multiview panel compares CAVERT with Generic LLM, Contract LLM, and Rules.

Transformations and Evaluation Protocol. Original retains the supplied execution log. Recode A consistently renames `event_id` and `actor_id`, updates their references, and reverses the alphabetical order of JSON keys. Recode B flattens the JSON hierarchy into a line-based log format with explicit causal pointers. Both preserve the recorded relationships and original reference labels.

Missing L1 removes direct prerequisite-delivery confirmations (`context_deliver`) and explicit handshake receipts, and sets `channel_complete` to false. Missing L2 starts from L1 and also removes pre-deadline state-retirement events (`effect_retirement`) and intermediate action confirmations. It additionally sets `causal_complete` and `complete_through_deadline` to false. These flags describe lost coverage, rather than evidence that an unrecorded event did not occur. Each removal view receives a reference judgment based on its remaining evidence.

Table A12: Diagnosis across execution models and evidence views.

Benchmark	Executor	n	Generic LLM		Contract LLM		Rules		**CAVERT**	
			Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
A. Cross-executor diagnosis: 720 executions										
MARBLE	DeepSeek	60	63.3	0.435	80.0	0.742	78.3	0.731	**86.7**	**0.828**
	Gemini	60	60.0	0.412	76.7	0.708	75.0	0.695	**83.3**	**0.792**
	Qwen	60	61.7	0.421	78.3	0.725	76.7	0.714	**85.0**	**0.810**
	Llama	60	66.7	0.468	83.3	0.785	81.7	0.768	**90.0**	**0.865**
CoffeeBench	DeepSeek	60	68.3	0.479	85.0	0.802	83.3	0.785	**90.0**	**0.865**
	Gemini	60	61.7	0.425	78.3	0.728	76.7	0.710	**85.0**	**0.812**
	Qwen	60	63.3	0.438	80.0	0.745	78.3	0.732	**86.7**	**0.830**
	Llama	60	65.0	0.452	81.7	0.762	80.0	0.748	**88.3**	**0.848**
MECoBench	DeepSeek	60	66.7	0.462	81.7	0.765	80.0	0.745	**88.3**	**0.846**
	Gemini	60	65.0	0.448	80.0	0.742	78.3	0.728	**86.7**	**0.828**
	Qwen	60	65.0	0.450	81.7	0.760	80.0	0.742	**88.3**	**0.845**
	Llama	60	63.3	0.432	80.0	0.738	78.3	0.722	**86.7**	**0.825**
All benchmarks / executors		720	64.2	0.435	80.6	0.748	78.9	0.738	**87.1**	**0.835**
B. Evidence transformations: 180 traces, five views										
Original		180	58.9	0.414	82.2	0.845	80.0	0.829	**87.8**	**0.895**
Recode A		180	56.7	0.399	80.6	0.833	80.0	0.829	**87.8**	**0.895**
Recode B		180	53.9	0.379	76.1	0.792	78.9	0.821	**86.7**	**0.886**
Missing L1		180	41.1	0.311	68.3	0.709	65.6	0.683	**78.3**	**0.801**
Missing L2		180	23.9	0.217	56.1	0.568	57.2	0.578	**66.1**	**0.661**

Acc. is percent; F1 is three-class macro-F1. Upper: 720 executions across three benchmarks and four executors. Lower: a separate 180-trace set under five evidence views; all metrics use the same records.

The link to the verdict rules is finding-specific. Each obligation is checked over its relevant channels and time interval. A false required coverage condition prevents that finding from establishing V. An earlier N rule may still apply; otherwise the judgment is U (Table A1). At trace level, V requires at least one finding with a certified window and a violated duty, even if other findings remain unresolved.

Methods are scored on the same instances and reference labels within each view, using the configurations in Table A5. Cross-view agreement measures prediction stability; within-view accuracy measures agreement with that view's reference judgments.

Table A13: E3: Joint prediction transitions from per-trace records.

View	Pred U	To U	From U	N/V flip	Same	Assert U
Generic LLM						
Original	0	0	0	0	180	18
Recode A	0	0	0	74	106	18
Recode B	0	0	0	73	107	18
Missing L1	0	0	0	78	102	38
Missing L2	0	0	0	70	110	76
Contract LLM						
Original	15	0	0	0	180	3
Recode A	15	3	3	51	123	3
Recode B	14	2	3	60	115	4
Missing L1	26	15	4	61	100	12
Missing L2	43	35	7	57	81	33
Strong rules						
Original	15	0	0	0	180	3
Recode A	15	0	0	0	180	3
Recode B	15	2	2	58	118	3
Missing L1	25	15	5	62	98	13
Missing L2	44	35	6	61	78	32
CAVERT						
Original	16	0	0	0	180	2
Recode A	16	0	0	0	180	2
Recode B	16	1	1	32	146	2
Missing L1	30	18	4	47	111	8
Missing L2	51	41	6	55	78	25

Transitions compare each view with Original on the same 180 traces. Pred U counts U outputs; To/From U counts N/V-to-U and U-to-N/V changes; N/V flip counts N-to-V or V-to-N changes; Same counts unchanged predictions. Assert U counts definite answers on that view's reference-U cases (18 in lossless views, 38 in L1, 76 in L2).

Table A14: Prediction stability under recoding and judgments after evidence removal.

Pattern	CAVERT	Rules	CAVERT example
Original to Recode B: paired predictions on 180 traces			
Unchanged prediction	146/180	118/180	—
Correct to wrong	18/180	32/180	`trace_marble_016`
Wrong to correct	16/180	30/180	`trace_marble_017`
Within Missing L2: prediction against the remaining-evidence reference			
Reference U, predicted U	51/76	44/76	—
Reference U, predicted N	25/76	32/76	`trace_coffeebench_047`
Reference V, predicted N	18/52	23/52	`trace_marble_004`
Reference N, predicted V	18/52	22/52	`trace_marble_054`

Recode B retains Original's reference labels. Missing L2 has 76 U, 52 V, and 52 N references; denominators are the corresponding class sizes. Its three error rows exhaust the observed L2 errors: 61 for CAVERT and 77 for Rules. Recoding transitions and L2 errors concern separate views of the same traces and are not added together.

D.3 Prediction Stability and Evidence Sufficiency

Missing L2 tests recognition of insufficient evidence and detection of violations supported by the retained records. Table A14 reports U-to-N and V-to-N errors alongside false violation predictions (N-to-V).

Table A15: Paired accuracy differences, CAVERT minus rules.

View	Difference (pp)	95% interval
Original	+7.8	[+5.0, +10.6]
Recode A	+7.8	[+5.0, +10.6]
Recode B	+7.8	[+4.4, +11.7]
Missing L1	+12.8	[+10.0, +15.6]
Missing L2	+8.9	[+6.1, +11.7]

10,000 bootstrap draws stratified by benchmark, conditional on the recorded 30-cluster partition of traces. Methods and views share draws.

Missing L2: judgments from the retained evidence

Rows are reference labels and columns are predictions, both ordered V,N,U:

$$M_{\text{CAVERT}} = \begin{pmatrix} 34 & 18 & 0 \\ 18 & 34 & 0 \\ 0 & 25 & 51 \end{pmatrix}, \qquad M_{\text{Rules}} = \begin{pmatrix} 29 & 23 & 0 \\ 22 & 30 & 0 \\ 0 & 32 & 44 \end{pmatrix}.$$

CAVERT's 25 U-to-N errors comprise five CoffeeBench and twenty MECoBench traces; its 18 V-to-N errors are all from MARBLE. Sixteen of these 18 violations were correctly predicted in the original view. For example, `trace_marble_004` retains reference V but changes from predicted V in L1 to N in L2. CAVERT correctly classifies 119/180 traces and Rules 103/180. These class-specific counts evaluate both evidence sufficiency and duty fulfillment.

D.4 Diagnosis under a Fixed Spending Limit

Shared diagnostic inputs and spending limit

We compare CAVERT with Contract LLM + review on the same 720 E3 executions, public logs, and task contracts. Each benchmark contains four executors and 60 executions per executor. Contract LLM + review extends the single-round Contract LLM with an initial judgment, contract-by-contract review, and revision. Both methods have a per-execution diagnostic spending limit of $B = \$0.0120$, selected on an independent development set and fixed before test evaluation. Stage 1 uses Qwen3.5-Flash; Stage 2 and all Contract LLM calls use DeepSeek-V4-Flash. The limit covers the entire diagnostic process, including retries; CAVERT's final verdict rules require no LLM call. The spending limit uses a fixed schedule per million input/output tokens: $\$0.14/\$0.28$ for DeepSeek-V4-Flash and $\$0.05/\$0.15$ for Qwen3.5-Flash.

Every execution remains in the evaluation denominator. Budget exhaustion or an invalid final response is recorded as a diagnostic failure, distinct from the semantic verdict U. A failure receives no accuracy credit and contributes a false negative for its reference class in N/V/U macro-F1. A valid U judgment instead denotes insufficient execution evidence.

CAVERT correctly diagnoses 627/720 executions, compared with 602/720 for Contract LLM + review (Table A16). Its accuracy advantage is +3.47 pp with a 95% interval of [+1.39, +5.56]; its macro-F1 advantage is +0.049 [+0.021, +0.076]. Both point estimates favor CAVERT in every benchmark (Table A17). The comparison extends the practical benefit of the criteria to a baseline with iterative contract review under the same spending limit. CAVERT averages 1.81 model calls per execution, versus 2.65 for the review baseline. Mean API cost is $\$0.0017$ for CAVERT and $\$0.0031$ for Contract LLM + review, with 95th-percentile costs of $\$0.0026$ and $\$0.0042$, respectively (Table A18). The diagnostic advantage thus persists with iterative review available to the baseline, while CAVERT uses fewer calls and lower API spending under the fixed price schedule.

For uncertainty estimates, 10,000 paired bootstrap replicates resample original task clusters within each benchmark. Each sampled cluster retains its executions, executor configurations, and both methods' predictions. Accuracy and three-class macro-F1 are recomputed from pooled predictions in each replicate. The 95% intervals are the 2.5th and 97.5th percentiles of paired differences. Pooled macro-F1 is computed across executions; it is not the mean of benchmark macro-F1 scores.

Table A16: Diagnosis under a common per-execution spending limit of $0.0120.

Benchmark	Method	Correct	Acc. (%)	Macro-F1	Calls
MARBLE	Contract LLM + review	198/240	82.50	0.778	2.64
MARBLE	**CAVERT**	**207/240**	**86.25**	**0.824**	**1.80**
CoffeeBench	Contract LLM + review	203/240	84.58	0.801	2.66
CoffeeBench	**CAVERT**	**210/240**	**87.50**	**0.839**	**1.82**
MECoBench	Contract LLM + review	201/240	83.75	0.779	2.65
MECoBench	**CAVERT**	**210/240**	**87.50**	**0.841**	**1.81**
Pooled	Contract LLM + review	602/720	83.61	0.786	2.65
Pooled	**CAVERT**	**627/720**	**87.08**	**0.835**	**1.81**

Each benchmark aggregates four executors $\times$ 60 executions. Calls are per-execution means over the complete diagnostic process, including format-repair attempts and retries. Both methods may finish below the spending limit. Single-round Contract LLM results are reported separately in Table 2.

Table A17: Paired performance differences under the common spending limit.

Scope	Δ Acc. (pp)	95% interval	Δ Macro-F1	95% interval
MARBLE	+3.75	[+0.42, +7.08]	+0.046	[+0.012, +0.080]
CoffeeBench	+2.92	[+0.00, +5.83]	+0.038	[+0.006, +0.070]
MECoBench	+3.75	[+0.83, +6.67]	+0.062	[+0.025, +0.099]
Pooled	+3.47	[+1.39, +5.56]	+0.049	[+0.021, +0.076]

Differences are CAVERT minus Contract LLM + review. Intervals use 10,000 paired task-cluster bootstrap replicates stratified by benchmark.

Table A18: API usage and cost across all 720 E3 executions.

Method	Diagnostic path	Calls	Mean cost	P95 cost
Generic LLM	Direct judgment	1.00	$0.0010	$0.0014
Rules	Deterministic checks	0.00	$0.0000	$0.0000
Contract LLM	Single-round contract judgment	1.00	$0.0014	$0.0021
Contract LLM + review	Initial judgment, review, revision	2.65	$0.0031	$0.0042
CAVERT	Scope filtering, causal-state inference, deterministic verdict	**1.81**	**$0.0017**	**$0.0026**

Calls and costs cover each execution's complete diagnostic process, including format-repair attempts and billed failed calls or retries. Means use all 720 executions; P95 is the 95th percentile of per-execution total cost. USD costs use recorded token consumption and the fixed price schedule in the text. Per-call output ceilings are not consumption measurements. Rules incur no model API cost.

E Execution Protocol and Stratification

E.1 Matched Blocks and Executor Configurations

E4 and E5 share 720 fault blocks but score complete executions and truncated probes separately. E5 reports 6,480 base nine-arm probes, 720 fixed-step Rules+G sensitivity probes, and 576 probes from four length configurations on 144 edge-case blocks. Step-0 logs are shared, so E4 and E5 reported run counts are not simply added.

E4 assigns 90 blocks per benchmark to each executor configuration. Config-A uses DeepSeek-V4-Flash, temperature 0.2, top-$p = 0.95$, a 2,048-token output ceiling, a 30-action budget, and a 180-second timeout. Config-B uses Gemini-2.5-Flash, temperature 0, top-$p = 1$, a 4,096-token output ceiling, a 50-action budget, and a 300-second timeout. Their controllers are the standard and extended variants of `cavert_controller_v4.2`. The original allocation has six arms: Native, Retry, Direct+G, Rules+G, CAVERT-Prompt, and CAVERT-Gate. Supplementary MemTX evaluation adds one arm on the same 720 blocks, retaining each block's executor configuration and resource ceilings. Table A19 and Figure 7 display the six intervention arms. The Native score of 27.8% in Table A21 belongs to E5's fixed-length probes, not E4's complete-execution endpoint.

Table A19: Matched complete-execution recovery with benchmark-specific endpoints.

			Baselines				**CAVERT**	
Benchmark	Endpoint	n/arm	Retry	Direct+G	Rules+G	MemTX	Prompt	Gate
Matched complete executions: 180 blocks per benchmark								
AppWorld	Success %	180	31.1	38.9	49.4	55.0	43.3	**67.8**
LongCLI	F2P+P2P %	180	12.2	18.9	28.3	32.8	22.2	**41.7**
MARBLE	Score %	180	46.7	53.9	64.4	72.2	57.8	**78.3**
CoffeeBench	Payout %	180	43.9	51.1	62.2	68.9	55.0	**76.1**
Equal-weight normalized mean		720	33.5	40.7	51.1	57.2	44.6	**66.0**

G: shared gate. All rows use two executor configurations, with 90 blocks each and 180 per arm (Appendix E). MemTX adds one arm on the same blocks. Payout is clipped normalization, not dollar income. The final row averages four normalized endpoint scores, not task-success probabilities across benchmarks. Paired differences appear in Table A20.

Table A20: Paired recovery score differences on the same 720 fault blocks.

Benchmark	MemTX − Rules+G	CAVERT-Gate − MemTX
AppWorld	$+5.56$ $[1.11, 10.00]$	$+12.78$ $[7.78, 17.78]$
LongCLI	$+4.44$ $[-0.56, 9.44]$	$+8.89$ $[3.89, 13.89]$
MARBLE	$+7.78$ $[3.33, 12.22]$	$+6.11$ $[1.67, 10.56]$
CoffeeBench	$+6.67$ $[2.22, 11.11]$	$+7.22$ $[2.78, 11.67]$
Equal-weight mean difference	$+6.11$ $[3.89, 8.33]$	$+8.75$ $[6.39, 11.11]$

Entries are differences in percentage points with 95% intervals, computed before rounding scores. The 10,000 percentile-bootstrap replicates resample matched blocks within each benchmark, keeping task, executor configuration, and seed paired across arms. Each aggregate replicate averages the four benchmark differences with equal weight. Intervals are pointwise.

Table A21: E5: Nine-arm suffix-probe results.

Arm	Macro-average score (%)
native	27.8
retry	33.5
generic prompt	36.9
rules prompt	42.5
direct prompt	43.1
cavert prompt	48.3
rules gate	51.1
direct gate	52.2
cavert gate	66.0

Matched executor limits refer to the action budget, model-output ceiling, and timeout assigned to a block. Diagnostic calls have their own per-call output ceilings (Table A5). Task-level token consumption, latency, and billed cost describe realized overhead and are reported separately in Table A28.

MemTX (Li et al., 2026) is a complete memory-system comparator, with belief commit, action gating, and cascading repair. Table A20 compares its recovery scores with Rules+G and CAVERT-Gate. For each of 10,000 percentile-bootstrap replicates, we resample matched blocks within each benchmark, preserving task, executor configuration, and seed across arms. The aggregate is the equal-weight mean of the four benchmark differences. CAVERT-Gate improves on MemTX in every benchmark, with an aggregate difference of +8.75 pp [6.39,11.11]. MemTX improves on Rules+G by +6.11 pp [3.89,8.33] overall; its LongCLI interval includes zero. These are comparisons of complete intervention configurations.

Table A22: Diagnosis–intervention interaction under fixed-length suffix evaluation.

720 matched blocks; four-benchmark mean				
Judge	Prompt (%)	Gate (%)	Gain (pp)	95% paired CI
Strong rules	42.5	51.1	+8.6	[5.2, 12.0]
Direct LLM	43.1	52.2	+9.1	[5.6, 12.6]
CAVERT	48.3	66.0	+17.7	[13.8, 21.6]

AppWorld, LongCLI, MARBLE, and CoffeeBench contribute 180 blocks each; judges share the gate. Gain is gate minus prompt. CAVERT versus rules interaction: +9.1 pp, 95% CI [4.8,13.4], $p < 0.001$. Scores average benchmark-specific normalized endpoints, not pooled success probabilities.

Table A23: Budget sensitivity and interference on normal tasks.

A. Budget sensitivity: 240 fault blocks

Actions / tokens	Generic Prompt	Rules+G	CAVERT+G
15 / 50k	28.3	41.7	54.2
30 / 100k	36.9	51.1	66.0
50 / 200k	40.8	55.4	71.3

B. Normal tasks: 480 uninjected blocks

Method	Score	Δ (pp)	95% CI	Non-inferiority
Native	88.5	+0.0	Reference	–
Rules+G	86.1	−2.4	[−4.1, −0.7]	Not established
MemTX (mature)	86.7	−1.8	[−3.4, −0.2]	Not established
MemTX (unmatured)	84.6	−3.9	[−5.6, −2.1]	Not established
CAVERT+G	87.8	−0.7	[−2.1, +0.7]	Supported

Scores are percentages. A reuses a matched-fault subset; action/token values are ceilings. B compares interventions with Native on the same 480 blocks. Non-inferiority requires the 95% interval's lower bound to exceed −2.5 pp (paired-block bootstrap; Appendix E.3).

E.2 Recovery Controls and Outcome Definitions

Native, retry, and generic-prompt controls accompany rules, direct LLM, and CAVERT crossed with prompt/gate. Arms share initial prefixes, action interfaces, and executor limits within a block. The gate intercepts writes before commitment.

Recovery endpoints and aggregate score

AppWorld uses binary official success; LongCLI uses joint F2P/P2P pass; MARBLE uses a continuous task score. CoffeeBench uses the normalization

$$s_{\mathrm{CB}} = \min(1, \max(0, \mathrm{Payout}/500)), \qquad \bar{s} = \tfrac{1}{4}\sum_{b=1}^{4} \bar{s}_b.$$

Payout is cumulative total earnings within a 30-environment-action scoring window, fixed across arms and executor configurations. The preregistered custom clipping threshold, $\mathrm{Payout}_{\mathrm{trunc}} = \$500$, uses the reported 95th-percentile threshold of historical market scenarios. All arms share this normalization and zero floor. This windowed earnings score is reported separately from cumulative net income; the executor's action ceiling controls the run budget rather than the scoring window.

E.3 Normal-Task Impact and Budget Sensitivity

Table A24 reports normal-task scores for four 120-block cohorts. CAVERT meets the −2.5 pp non-inferiority criterion for the 480-block aggregate; the MARBLE stratum's lower bound equals the margin. On the same blocks, mature and unmatured MemTX score 86.7% and 84.6%, respectively.

Table A24: Normal-task scores and paired differences from Native, by benchmark.

A. Scores (%); MemTX uses mature initialization

Cohort	Blocks	Native	Rules+G	MemTX	CAVERT+G
AppWorld	120	90.0	87.5	88.3	89.2
LongCLI	120	85.0	82.5	83.3	84.2
MARBLE	120	89.2	86.7	87.5	88.3
CoffeeBench	120	89.8	87.8	87.8	89.4
Equal-weight mean	480	88.5	86.1	86.7	87.8
Historical τ^2	51	88.2	85.8	–	87.5

B. Differences from Native (pp), with 95% intervals

Cohort	MemTX − Native	CAVERT+G − Native
AppWorld	-1.7 $[-3.3, +0.0]$	-0.8 $[-2.3, +0.7]$
LongCLI	-1.7 $[-3.4, +0.1]$	-0.8 $[-2.4, +0.8]$
MARBLE	-1.7 $[-3.4, +0.1]$	-0.9 $[-2.5, +0.7]$
CoffeeBench	-2.0 $[-3.5, -0.5]$	-0.4 $[-1.8, +1.0]$
Equal-weight mean	-1.8 $[-3.4, -0.2]$	-0.7 $[-2.1, +0.7]$
Historical τ^2	–	-0.7 $[-2.1, +0.7]$

Four 120-block cohorts form the 480-block aggregate. The historical 51 are separate; no MemTX task score is reported for that cohort. Its permitted-action comparison appears in Table A28. Normal-task scores and action false-block rates have different denominators.

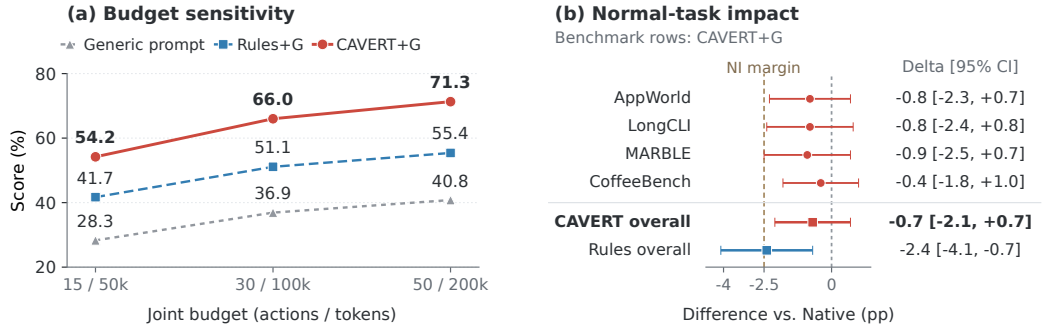

Figure A1: Budget sensitivity (a, 240 fault blocks) and paired normal-task differences from Native (b, 480 blocks). Benchmark rows show CAVERT+G with 95% CIs. Dashed lines mark zero and the -2.5 pp margin. Protocol: Tables A23 and A24.

Each is compared with Native: -1.8 pp $[-3.4, -0.2]$ and -3.9 pp $[-5.6, -2.1]$. Both intervals cross the non-inferiority margin.

The 51-block τ^2-bench (Barres et al., 2025) action-audit cohort is separate. Of its 750 permitted actions, Rules+G blocks 43, MemTX 29, and CAVERT+G 12 (Table A28). These action-level rates are distinct from four-benchmark task scores.

For normal-task uncertainty, we resample the 480 matched blocks with replacement 10,000 times, retaining each block's arm scores together. The 2.5th and 97.5th percentiles of the mean CAVERT-minus-Native difference give $[-2.1, +0.7]$ pp around -0.7 pp. Non-inferiority requires the lower bound to exceed the predeclared -2.5 pp margin.

Budget tiers jointly change action ceilings (15, 30, 50) and token ceilings (50k, 100k, 200k). They therefore measure combined budget sensitivity, not an isolated token effect.

Table A25: Recovery across execution models and supplementary cohorts.

				Baselines			**CAVERT**	
Benchmark	Executor	Endpoint	n/arm	Retry	Direct+G	Rules+G	Prompt	Gate
Cross-backbone recovery								
AppWorld	DeepSeek	Recovery %	120	25.8	34.2	51.7	36.7	**68.3**
	Gemini		120	26.7	35.0	52.5	37.5	**69.2**
	Qwen		120	20.8	29.2	45.8	30.8	**61.7**
	Llama		120	22.5	30.8	47.5	32.5	**64.2**
LongCLI	DeepSeek	F2P+P2P %	60	6.7	13.3	28.3	15.0	**43.3**
	Gemini		60	6.7	13.3	28.3	15.0	**43.3**
	Qwen		60	5.0	11.7	25.0	13.3	**38.3**
	Llama		60	6.7	13.3	26.7	15.0	**40.0**
Original and extended cohorts: native endpoints								
AppWorld	4-model	Success %	480	18.33	25.00	40.83	27.50	**54.17**
LongCLI	4-model	F2P+P2P %	240	6.25	12.92	27.08	14.58	**41.25**
MARBLE	Llama	Score %	64	54.2	67.8	76.5	73.5	**88.5**
CoffeeBench	DeepSeek	Income (USD)	18	1,240	2,420	3,580	3,150	**4,680**
EntCollabBench	DS/Gemini	Recovery %	24	33.3	–	–	54.2	**79.2**
PAST-Bench	Qwen/Llama	Score [0,1]	90	0.583*	–	–	–	**0.867***
EntCollab‡	DeepSeek	Workflow %	180	37.8	48.3	58.9	45.6	**72.2**

G: shared gate. AppWorld recovery flags and official success are distinct endpoints. * PAST uses persistence/controller interfaces (Table A27). EntCollab (24 groups): DeepSeek-V4-Flash/Gemini-2.5-Flash; ‡: independent extension. Dollar income is unclipped. Cohort settings: Appendix E.4.

E.4 Supplementary Cohorts, Outcomes, and Costs

The supplementary AppWorld cohort contains 40 task families, four models, and three seeds (480 blocks per arm). LongCLI contains 20 tasks, four models, and three seeds (240 blocks per arm). Both allow 15 interaction steps: 11 prefix steps and four recovery steps. This setting is separate from the budget sweep. AppWorld scores recovery sequencing, state postconditions, and official success separately: CAVERT+G obtains 316/480 recovery flags and 260/480 official successes. Paired resampling retains AppWorld families and repository-aware LongCLI groups with all models, seeds, and arms together. MARBLE uses task groups and CoffeeBench uses worlds. The CoffeeBench dollar-income panel reports unclipped income, separately from the normalized score panel.

Our EntCollabBench (Yu et al., 2026a) cohort has 24 recovery groups, six per fault family, within the original 36 semantic groups. Success requires official task passage and a final database state matching the reference. The 24 groups across three arms give 72 runs, spanning DeepSeek-V4-Flash and Gemini-2.5-Flash. Retry, prompt, and gate succeed on 8/24, 13/24, and 19/24 groups, respectively. The gate-minus-prompt difference is +25.0 pp, with 11 gate-only and five prompt-only successes. A two-sided exact McNemar test gives $p = 0.2101$; this small-cohort difference is not statistically significant at 0.05.

Our PAST-Bench (Xue et al., 2026) evaluation measures the fraction of subgoals passed over 90 episodes per arm (30 near, 60 far), five families, and three seeds. No persistence, generic persistence, structured records, and contract control score 0.425, 0.583, 0.742, and 0.867. State access and control authority vary jointly. We aggregate episodes and seeds within families, then resample the five families 20,000 times. Models are fixed within families.

The action panel reuses the 750 permitted actions rather than adding independent task blocks. Token increases use a common Native mean of 12,000 tokens per task. MemTX uses 14,220 tokens and 22.3 seconds per task; CAVERT+G uses 14,950 tokens and 23.5 seconds. Their AppWorld/LongCLI costs are $0.044 and $0.042 per task under DeepSeek-V4-Flash billing, including prompt-cache discounts. Token totals alone therefore do not determine billed cost.

Table A26: Five-arm supplementary recovery: native task outcomes and suffix counts.

A. Native multi-agent complete tasks				
	MARBLE: 64 tasks		CoffeeBench: 18 worlds	
Recovery method	Task score ↑	Violations (%) ↓	Income (USD) ↑	Violations (%) ↓
Standard retry	0.542	31.6	1,240	42.3
Fresh-read recovery	0.615	25.2	1,850	33.5
Direct diagnosis + gate	0.678	18.4	2,420	24.8
CAVERT prompt-only	0.735	11.2	3,150	15.6
CAVERT + gate	**0.885**	**1.8**	**4,680**	**2.3**

B. Stateful recovery suffixes: counts					
	AppWorld: 480 blocks/arm			LongCLI: 240 blocks/arm	
Recovery method	Recovery flag	Postcondition	Success	F2P+P2P	Regressions
Standard retry	115	104	88	15	41
Fresh-read recovery	137	124	108	23	34
Direct diagnosis + gate	155	136	120	31	23
CAVERT prompt-only	165	152	132	35	18
CAVERT + gate	**316**	**292**	**260**	**99**	**4**

Direct uses the same gate; prompt-only uses CAVERT diagnosis without blocking. A: violations per assessed obligation. B: 40 AppWorld families / 20 LongCLI tasks × 4 models × 3 seeds. AppWorld scores recovery sequencing, state postconditions, and official success separately. F2P+P2P requires repair without test regressions; fewer regressions is better.

Table A27: Enterprise recovery and cross-session scores under their original intervention interfaces.

Panel / endpoint	Reference	Record	Controller	$\Delta_{\mathrm{C-R}}$ [95% CI]
EntCollabBench: recovery	0.333	0.542	0.792	+0.458 [0.250, 0.667]
PAST-Bench: cross-session score	0.583	0.742	0.867	+0.284 [0.183, 0.385]

$\Delta_{\mathrm{C-R}}$ is Controller minus Reference before rounding, in rate/score units. EntCollabBench columns: Retry, CAVERT prompt, CAVERT Gate (8/24, 13/24, 19/24). PAST-Bench columns: Generic Persistence, Root/Trigger Record, Contract Controller; 90 episodes per arm and 20,000 paired bootstrap draws over five families.

Table A28: Action false blocks, execution overhead, and monetary costs, with separate cohorts and units.

A. Historical τ^2-bench: 750 permitted actions

Method	False blocks / permitted actions	Rate	95% CI
Native	0/750	0.0%	–
Rules+G	43/750	5.7%	[4.2%, 7.5%]
MemTX	29/750	3.9%	[2.6%, 5.4%]
CAVERT+G	12/750	1.6%	[0.8%, 2.5%]

B. Long-horizon execution: per-task means

Method	Tokens	Token increase	Latency (s)	Added latency (s)
Native	12,000	Reference	18.5	Reference
Rules+G	13,200	+10.0%	19.8	+1.3
MemTX	14,220	+18.5%	22.3	+3.8
CAVERT+G	14,950	+24.6%	23.5	+5.0

C. AppWorld + LongCLI: DeepSeek-V4-Flash billing

Unit	Native	Rules+G	MemTX	CAVERT+G
USD / task	0.030	0.035	0.044	0.042

A: separate from the 720 fault and 480 normal blocks; – denotes no interval. B: per-task means on a separate long-horizon cohort, relative to Native. C: AppWorld/LongCLI billed costs including prompt-cache discounts.

E.5 LIMITATIONS

Our claims are limited to the evaluated contracts, evidence factors, and log transformations; natural-log frequencies characterize only the sampled sources. CAVERT recovers semantic relationships from existing logs rather than observing them natively, so diagnostic quality remains bounded by extraction quality. The recovery gate acts before commitment; repairing committed external effects requires compensation mechanisms beyond our current scope.