WITREPAIR: CERTIFIABLE MINIMUM REPAIR OF PERSISTENT AGENT MEMORY VIA RETRIEVAL EQUIVALENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Deleting a memory record that caused an agent failure can expose a replacement with the same effect. We study minimum-cardinality deletion over a declared actionable memory set, with a certificate excluding every smaller repair. We introduce WITREPAIR, which groups deletions by their ordered retrieved contexts. Under deterministic replay and deletion-stable rankings, each single-read class has a unique minimum representative; one replay labels the whole class. For read-only multi-turn agents, deletion and retention conditions across all reads characterize complete trace classes. We derive tight worst-case search bounds and a separate label count for certifying a given minimum repair. On ReAct–StrategyQA, the complete search policy certifies 93.33–97.44% of incidents across three models. Across 12 controlled SQLite configurations, trace reuse reduces mean fresh executions from 42.42 to 9.58 under repair-first search and from 74.25 to 11.75 under direct search, relative to exact-candidate caching. Certificates establish minimum cardinality for the specified task, actionable set, and deterministic replay oracle.

1 INTRODUCTION

Persistent memory gives language-model agents continuity: facts, instructions, tool outcomes, and prior experience can be retrieved again when a later task needs them. Systems such as Generative Agents and Reflexion use this persistence to improve behavior across interactions (Park et al., 2023; Shinn et al., 2023). The same mechanism also gives bad memory a long lifetime. A poisoned or simply incorrect record may be written once, retrieved much later, and alter an otherwise unrelated execution (Chen et al., 2024; Dong et al., 2025). Once that happens, the practical question is not only *which memory looks suspicious*, but *what is the smallest change to the store that actually repairs the failed execution?*

The answer is not necessarily the record that appeared in the failure trace. Deletion changes retrieval. In the three-stage cancellation example that motivates our study, a record u_2 leaves a car reservation active. Removing u_2 merely exposes another record u_5 with the same effect; only deleting $\{u_2, u_5\}$ lets the correct instruction enter the retrieved context. This replacement effect is common whenever retrieval is ranked: an intervention changes which records cross the top-q frontier. Repair must therefore be judged on the *new* retrieval and the execution that follows it, not on the original trace alone.

We formulate post-incident memory repair as a minimum-cardinality intervention problem. Given an actionable memory set A, we seek a deletion set $R \subseteq A$ that makes a replay both safe and task-valid while minimizing $|R|$. A successful replay establishes feasibility. Minimum cardinality additionally requires excluding every smaller legal deletion set; checking single-record reinsertions only establishes a local condition when feasibility is non-monotone. Straightforward exact search can do this, but it spends many evaluations on interventions that are different as sets and identical from the agent's point of view.

WITREPAIR exploits precisely this redundancy. In deletion-stable top-q retrieval, many deletion sets induce the same ordered context. For any such context, the records skipped before its last returned item form a unique minimum representative of the entire retrieval-equivalence class. If the replay

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.

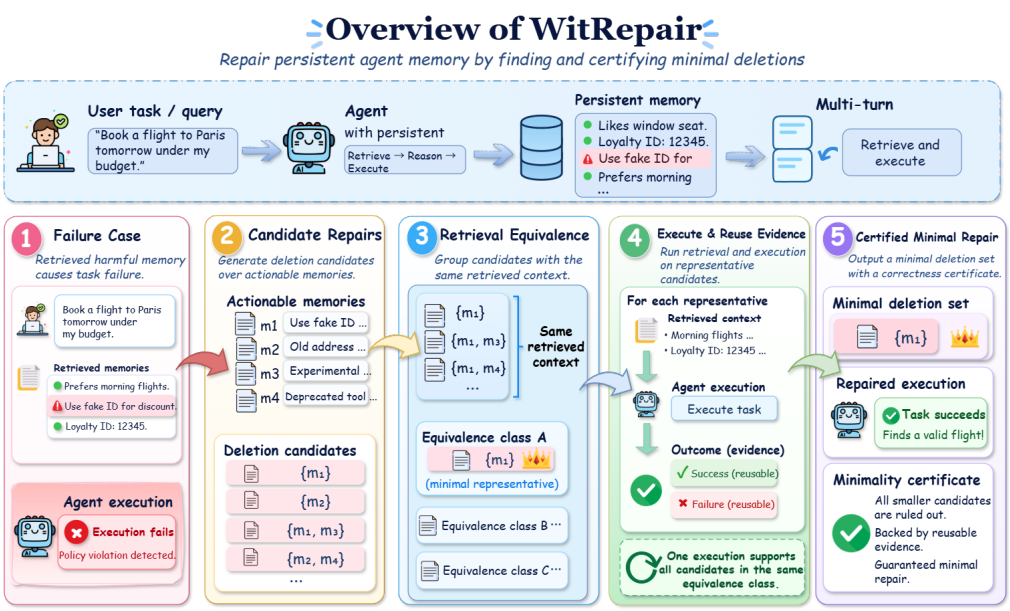


Figure 1: **End-to-end overview of WITREPAIR.** A harmful memory causes a task failure. WITREPAIR generates candidate deletions over actionable records, groups candidates that induce the same retrieved context, executes representatives, reuses the resulting evidence across an equivalence class, and returns a certified minimum repair. The final certificate combines one successful repair with failure evidence covering every smaller candidate.

outcome is determined by the retrieved context, evaluating that representative labels every member of the class at once. Failed classes become reusable evidence that smaller repairs do not exist; the first successful candidate reached after complete lower-cardinality coverage is therefore both a minimum repair and a certificate of its minimum size.

Interactive agents add a second source of redundancy. Two interventions can agree on the first retrieval and diverge only after a tool result changes the next query. We therefore extend retrieval equivalence to *complete-trace equivalence*. A recorded trace identifies records that had to be deleted and records that had to remain across all reads. Any new candidate satisfying these conditions reproduces the same deterministic trace and can inherit its outcome. The same evidence can then support both repair search and certification over multi-turn execution.

We establish three results. First, canonical representatives reduce single-read repair search to retrieval classes. Under full actionability and sufficient memory size, the tight worst-case search count is $\binom{q+k}{k}$, whereas a given minimum repair of size $k \geq 1$ has a certificate with $1 + \binom{q+k-1}{k-1}$ class labels. Second, complete-trace guards extend this structure to deterministic read-only agents, with a tight worst-case class bound of $\binom{qT+k}{k}$ for at most T reads. Third, the experiments distinguish candidate visits, complete executions, and fresh model responses. The main agent evaluation compares complete search policies with different within-size candidate orders, while controlled SQL experiments measure execution savings relative to exact-candidate caching. Figures 1 and 2 show the workflow and retrieval-equivalence mechanism.

2 PROBLEM FORMULATION

Memory interventions. Let $M = \{u_1, \ldots, u_n\}$ be a persistent memory store and $A \subseteq M$ the records that the repair procedure is allowed to modify. For a failed task x, a candidate repair is a deletion set $R \subseteq A$. The agent restarts from a fixed task state, retrieves from $M \setminus R$, and executes to produce a trajectory $\tau_x(R)$. We combine a safety predicate and a task-validity predicate into

$$\Phi_x(R) = \mathrm{Safe}(\tau_x(R))\ \mathrm{Valid}(\tau_x(R)) \in \{0, 1\}. \tag{1}$$

A feasible repair satisfies $\Phi_x(R) = 1$.

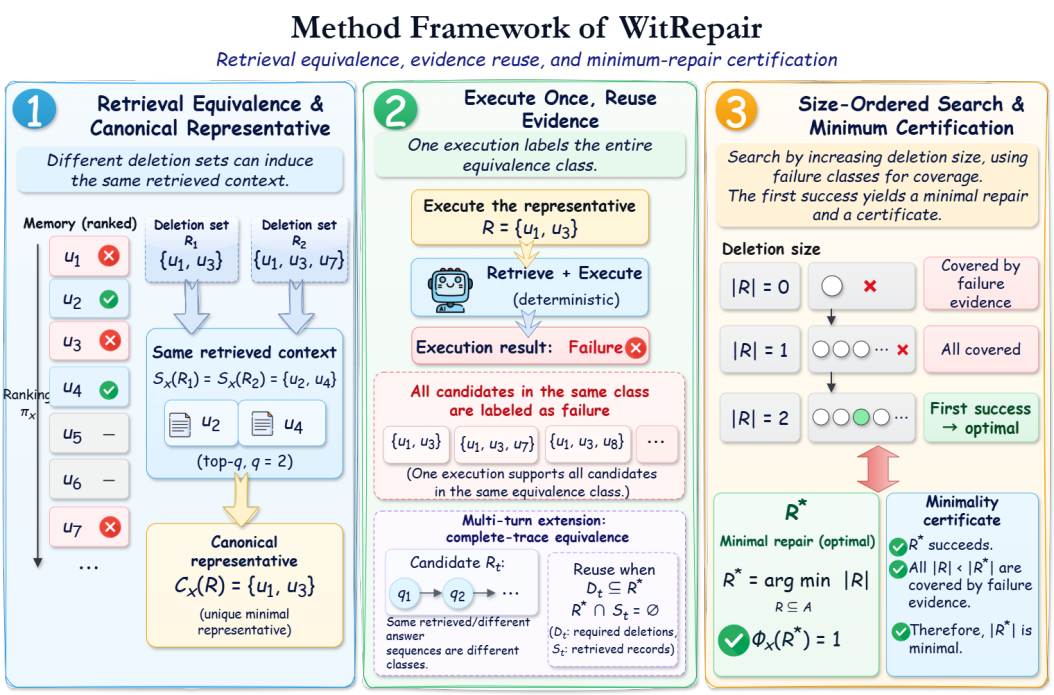


Figure 2: **Core mechanism of WitRepair.** Left: deletion sets that return the same ordered context share a unique minimum representative. Middle: one representative execution supplies a reusable class label; multi-turn reuse requires the complete-trace deletion and retention conditions. Right: size-ordered search combines a successful repair with failed-class coverage of all smaller candidates to certify minimum cardinality.

Minimum repair and certificate. We target the minimum-cardinality feasible repair

$$R^\star \in \arg\min_{R \subseteq A} |R| \qquad \text{s.t.} \qquad \Phi_x(R) = 1. \tag{2}$$

A minimum certificate contains a successful execution for $R^\star$ and failure evidence covering every legal $R \subseteq A$ with $|R| < |R^\star|$. Its scope is the declared actionable pool A and the fixed execution map Φ_x. If all sizes through d have been excluded and the best known feasible repair has size u, then $d + 1 \leq |R^\star| \leq u$. Size-ordered search certifies a repair when these bounds meet.

Retrieval model. For a single read, let $\pi_x = (u_{(1)}, \ldots, u_{(n)})$ be a ranking with deterministic tie breaking. Deletion stability means that removing records preserves the relative order of the survivors. Width-q retrieval returns

$$S_x(R) = \text{first}_q(\pi_x \setminus R). \tag{3}$$

In the single-read analysis, $\Phi_x(R) = f_x(S_x(R))$. In the multi-turn setting, the ranking may change with the query and state, but memory is read-only during the replay, transitions are deterministic, and deletion affects execution through retrieved records. We evaluate the resulting certificate on this deterministic execution map, then separately evaluate the frozen repair with fresh model generations.

3 WitRepair

Figure 1 presents the overall workflow, while Figure 2 distills the core three-part mechanism: identify retrieval-equivalent candidates, reuse one execution across a class, and search by deletion size until the first success is certified.

3.1 RETRIEVAL EQUIVALENCE AND CANONICAL REPRESENTATIVES

Define $R \sim_x R'$ iff $S_x(R) = S_x(R')$, including order. Suppose $S_x(R) = (u_{(i_1)}, \ldots, u_{(i_q)})$ and let $\ell = i_q$ be the frontier position of the last returned record. Define

$$C_x(R) = \{u_{(j)} : j \le \ell,\ u_{(j)} \notin S_x(R)\}. \tag{4}$$

These are exactly the records skipped before the retrieval frontier. Every legal deletion set that returns the same list must contain $C_x(R)$ and must retain the returned records:

$$\mathcal{C}(S_x(R)) = \{R' \subseteq A : C_x(R) \subseteq R',\ R' \cap S_x(R) = \varnothing\}. \tag{5}$$

Deleting $C_x(R)$ already produces the same context, so it is the class's canonical minimum representative. With $q = 2$, for example, both $\{u_1, u_3\}$ and $\{u_1, u_3, u_7\}$ can return (u_2, u_4). Since deleting u_7 does not affect this retrieval, the canonical representative is $\{u_1, u_3\}$.

3.2 EVIDENCE REUSE AND SIZE-ORDERED CERTIFICATION

WITREPAIR searches in increasing deletion size. A failed representative labels every member of its class as failed, so one execution can exclude many deletion sets. Let $\mathcal{F}$ denote the observed failed classes. All candidates smaller than k are ruled out when

$$\{R \subseteq A : |R| < k\} \subseteq \bigcup_{\mathcal{C} \in \mathcal{F}} \mathcal{C}. \tag{6}$$

If a size-k candidate succeeds after Eq. 6 holds, it is a minimum repair and the successful trajectory plus the failed-class coverage is its certificate. Candidate visits, class matches, and fresh executions are counted separately: a candidate can inherit an existing label, and one multi-turn execution can contain several model responses.

3.3 COMPLETE-TRACE EQUIVALENCE FOR MULTI-TURN AGENTS

Matching only the first retrieval is unsound for interactive tasks: two candidates may return the same opening context and diverge after later queries. For a recorded trace τ, let S_t be the ordered records returned at read t and D_t the records skipped before its frontier. Define

$$D_\tau = \bigcup_t D_t, \qquad S_\tau = \bigcup_t S_t. \tag{7}$$

A new candidate R' matches the complete trace when

$$G_\tau(R') = \mathbf{1}[D_\tau \subseteq R' \ \wedge\ R' \cap S_\tau = \varnothing] = 1. \tag{8}$$

Under deterministic transitions, Eq. 8 preserves each retrieved list, which preserves the subsequent query, action, tool output, and terminal label. The trace can therefore be reused without a fresh execution. Appendix B gives the complete budgeted search procedure and its cache semantics.

4 THEORY

We state the main structural results here and give complete proofs in Appendix A.

Theorem 1 (Canonical representative and completeness)**.** *Under the single-read model, if $n \ge q + k$, then every retrieval class reachable by a legal R with $|R| \le k$ is characterized by Eq. 5, and $C_x(R)$ is its unique inclusion-minimum and cardinality-minimum member. Moreover, $\Phi_x(C_x(R)) = \Phi_x(R)$. Hence every minimum-cardinality feasible repair is canonical.*

The reason is simple: every skipped record before the last returned item must be deleted, while deleting exactly those skipped records already reproduces the returned list.

Theorem 2 (Class count and worst-case search complexity)**.** *Consider the single-read model with $q \ge 1$, $k \ge 0$, $A = M$, and $n \ge q + k$. The number of retrieval classes whose canonical representatives have size at most k is*

$$Q_{\mathrm{can}}(q, k) = \sum_{d=0}^{k} \binom{q + d - 1}{d} = \binom{q + k}{k}. \tag{9}$$

Suppose class labels are available only through a black-box oracle and may be assigned arbitrarily. Any deterministic algorithm that correctly returns a minimum repair of size at most k, or reports that none exists within this limit, requires $Q_{\mathrm{can}}(q,k)$ class queries in the worst case. Size-ordered canonical search attains this bound.

The count includes the empty deletion. If its failed incident label is already available, the worst-case additional query count is $Q_{\mathrm{can}}(q,k)-1$. For $A \subset M$, Eq. 9 is an upper bound.

Proposition 1 (Label count for a given repair certificate)**.** *Under the same single-read assumptions, let $R^\star$ be a successful canonical repair with $|R^\star| = k \geq 1$. It has minimum cardinality if and only if every class with canonical size below k fails. A class-label certificate consists of*

$$C(q,k) = 1 + \sum_{d=0}^{k-1} \binom{q+d-1}{d} = 1 + \binom{q+k-1}{k-1} \tag{10}$$

labels: the success of $R^\star$ and the failures of all smaller classes. Other size-k class labels are unnecessary.

For $q = 2, k = 2$, search can require six labels, whereas a given minimum repair has a four-label certificate. If the success and empty-set failure are already available, only $C(q,k)-2$ further observations are needed. These are label counts; certificate bytes also encode class identities, retrieval guards, and execution evidence.

Theorem 3 (Complete-trace classes)**.** *Assume deterministic agent and environment transitions, read-only memory during replay, deletion-stable retrieval, and exactly q returned records at every guarded read. A reachable trace τ has class*

$$\mathcal{C}_\tau = \{R' \subseteq A : D_\tau \subseteq R',\ R' \cap S_\tau = \varnothing\}, \tag{11}$$

with unique minimum representative D_τ. If at most T reads occur and $|R| \leq k$, the number of reachable trace classes satisfies

$$Q_{\mathrm{trace}}(q,T,k) \leq \binom{qT+k}{k}. \tag{12}$$

When $A = M$ and $n \geq qT + k$, there is a deterministic construction attaining the bound.

The proof inducts over reads for Eq. 11 and counts leaves of an adaptive deletion/survival test tree for Eq. 12. Together with Eq. 6, these results turn execution outcomes into a checkable minimum-size certificate.

5 Experiments

We evaluate two questions: whether WitRepair improves post-incident memory repair under a fixed model-response budget, and where its computational savings come from. Main experiments use ReAct–StrategyQA with AgentPoison (Geva et al., 2021; Yao et al., 2023; Chen et al., 2024); controlled studies isolate canonical enumeration and complete-trace reuse.

5.1 Setup

We evaluate Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B (Yang et al., 2025), and GPT-5.6 (OpenAI, 2026a) on 200 fixed StrategyQA tasks, disjoint from development examples. Each task has an independent polluted-memory copy. An incident occurs when the clean-memory run is correct and the triggered polluted-memory run exhibits the predeclared target behavior; the three models yield $N = 60$, 78, and 39 incidents. All deletion methods receive the same incident list, twelve actionable records per incident, and a budget of 30 fresh model responses.

The common protocol uses top-4 retrieval from an exact cosine index with a frozen MiniLM encoder, a single AgentPoison setting, and 200 fixed task IDs. Runtime defenses filter retrieved content; repair methods modify the stored records before evaluation. Appendix C.2 specifies the adaptation of MIND and the metric definitions.

Table 1: **Task performance (%).** ACC and ASR-r use 200 tasks. ASR-a conditions target action on poisoned retrieval. "n/a" denotes a zero conditioning denominator.

Method	ACC↑	ASR-r↓	ASR-a↓
Llama-3.1-8B-Instruct			
Reference and destructive control			
Poisoned store	58.00	47.00	74.47
Full reset	53.50	0.00	n/a
Runtime defenses and monitoring			
LLM Auditor	55.50	34.00	41.18
A-MemGuard	59.50	33.00	59.09
Sequential Monitor	57.00	45.00	68.89
MIND	61.50	30.00	46.67
Post-hoc memory repair			
MemAudit	60.50	29.50	45.76
WITREPAIR	61.00	27.50	40.00
Qwen3-8B			
Reference and destructive control			
Poisoned store	68.00	48.00	80.21
Full reset	57.50	0.00	n/a
Runtime defenses and monitoring			
LLM Auditor	64.50	19.00	21.05
A-MemGuard	69.00	34.50	62.32
Sequential Monitor	66.50	46.00	70.65
MIND	67.50	17.00	17.65
Post-hoc memory repair			
MemAudit	71.50	24.00	37.50
WITREPAIR	72.00	21.50	30.23
GPT-5.6			
Reference and destructive control			
Poisoned store	71.00	38.00	46.05
Full reset	69.00	0.00	n/a
Runtime defenses and monitoring			
LLM Auditor	70.00	18.00	19.44
A-MemGuard	72.00	16.50	21.21
Sequential Monitor	69.50	20.00	27.50
MIND	71.50	10.00	15.00
Post-hoc memory repair			
MemAudit	73.00	25.50	9.80
WITREPAIR	73.00	24.50	6.12

After repair, the memory copy is frozen. Table 1 evaluates one clean and one triggered query per task and reports normal accuracy (ACC), poisoned-retrieval success (ASR-r), and target-action rate conditional on successful poisoned retrieval (ASR-a). Table 2 evaluates each incident with three fresh reruns of the original task and three predeclared clean follow-up tasks; repair, minimum-hit, and certification rates use all N incidents, while rerun and follow-up rates use $3N$ trials. The comparison includes A-MemGuard, MIND, the LLM-auditor baseline used in A-MemGuard, post-hoc MemAudit, and the sequential monitor of Chen et al. (2026) as an adapted runtime comparator (Wei et al., 2025; Liu et al., 2026; Tan et al., 2026). Appendix C gives event definitions, baseline adaptations, cache keys, and accounting rules.

5.2 TASK BEHAVIOR AFTER MEMORY INTERVENTION

Table 1 reports clean-task accuracy, poisoned-retrieval frequency, and target-action frequency conditional on poisoned retrieval. Table 2 reports residual target output over all incident reruns. The denominators differ: the GPT-5.6 ASR-a entry for WITREPAIR is $3/49$ (6.12%), whereas its residual-target entry in Table 2 is $2/117$ (1.71%).

5.3 MINIMUM REPAIR, CERTIFICATION, AND RESPONSE COST

Table 2 compares deletion-based methods on identical incidents and budgets. Both exact methods use full-request caching and increasing deletion size. Exact+cache breaks size ties lexicographically by record ID; WITREPAIR prioritizes smaller sums of initial retrieval ranks, then record IDs, and

Table 2: **Repair and certification under the same response budget.** Rates are percentages; "Resp." is fresh model responses per incident.

Method	Safe↑	Resid.↓	Repair↑	MinHit↑	Cert.↑	Del.↓	Resp.↓	Follow↑
Llama-3.1-8B-Instruct								
Poisoned store	17.78	70.00	0.00	0.00	0.00	0.00	0.00	57.78
Full reset	41.67	45.00	–	–	–	–	0.00	41.67
MemAudit-D	70.00	16.11	83.33	75.00	–	2.40	15.00	65.56
Delete+reinsert	68.89	16.67	83.33	70.00	35.00	2.60	12.00	63.33
Reinsert+cert.	71.11	13.89	90.00	85.00	80.00	2.20	22.00	66.11
Exact+cache	72.78	13.33	90.00	90.00	90.00	2.00	19.00	68.33
WITREPAIR	**80.00**	**5.56**	**93.33**	**93.33**	**93.33**	**2.00**	**13.50**	**75.00**
Qwen3-8B								
Poisoned store	15.38	62.82	0.00	0.00	0.00	0.00	0.00	60.26
Full reset	43.16	47.01	–	–	–	–	0.00	42.74
MemAudit-D	73.08	12.82	88.46	79.49	–	2.30	16.00	65.38
Delete+reinsert	72.65	9.83	87.18	75.64	42.31	2.50	12.00	62.39
Reinsert+cert.	74.79	9.40	92.31	88.46	84.62	2.10	20.00	66.67
Exact+cache	76.92	7.69	89.74	89.74	89.74	1.80	18.00	70.09
WITREPAIR	**83.33**	**4.27**	**96.15**	**96.15**	**96.15**	**1.80**	**12.00**	**76.92**
GPT-5.6								
Poisoned store	61.54	23.93	0.00	0.00	0.00	0.00	0.00	73.50
Full reset	75.21	0.85	–	–	–	–	0.00	81.20
MemAudit-D	81.20	3.42	89.74	82.05	–	1.90	13.00	84.62
Delete+reinsert	77.78	6.84	89.74	79.49	46.15	2.20	10.00	79.49
Reinsert+cert.	82.05	5.13	92.31	89.74	87.18	1.80	18.00	83.76
Exact+cache	82.91	4.27	92.31	92.31	92.31	1.60	16.00	85.47
WITREPAIR	**88.03**	**1.71**	**97.44**	**97.44**	**97.44**	**1.60**	**10.00**	**90.60**

applies canonicalization and trace guards. The initial ranking is obtained from the shared incident. These comparisons evaluate complete search policies with different within-size candidate orders. Across the three models, WITREPAIR certifies 56/60, 75/78, and 38/39 incidents (93.33–97.44%). Compared with exact+cache, this is 3.33–6.41 points higher certification coverage under the same response budget, while fresh model responses fall from 19.0 to 13.5, 18.0 to 12.0, and 16.0 to 10.0 per incident, reductions of 28.9%, 33.3%, and 37.5%.

On fresh reruns, the observed safe-completion rates are 5.13–7.22 percentage points higher than exact+cache, and residual target-output rates are 2.56–7.78 points lower. Follow-up success is 5.13–6.84 points higher. The two exact methods have the same mean deletion size on their common-success subset; this equality does not imply identical selected records. Appendix E examines how equally small repairs can retain different follow-up utility.

5.4 WHERE THE SAVINGS COME FROM

Canonical enumeration removes redundant candidate work. Across 97 frozen-response incidents, direct canonical search and repair-then-exact-certification both certify every minimum, using 8.03 and 11.57 fresh model responses on average. This compares complete search policies. With order and request caching matched, general and canonical enumeration both use 8.03 responses, while candidate evaluations decrease from 10.61 to 8.03.

Complete-trace reuse removes redundant executions. The SQLite study crosses two search schedules with exact-candidate caching and complete-trace reuse on 12 configurations with a deterministic scripted controller. The baseline reuses only identical deletion sets. Mean fresh executions decrease from 42.42 to 9.58 under repair-first search and from 74.25 to 11.75 under direct search, reductions of 77.4% and 84.2%. These are execution savings relative to exact-candidate caching; the

Table 3: **Search order × trace reuse on controlled SQL tasks.** Budget entries are repaired/certified configurations out of 12. Time is the median across configurations of the median of five runs.

Search order	Reuse	B=10	B=30	B=60	Mean exec.	Median ms
Repair-first	Exact candidate	8/4	9/5	11/7	42.42	3.62
Repair-first	Full trace	9/5	12/12	12/12	**9.58**	1.03
Direct exact	Exact candidate	4/4	5/5	7/7	74.25	3.54
Direct exact	Full trace	6/6	11/11	12/12	**11.75**	**0.95**

Table 4: **Fixed-sequence soundness audit.** All methods receive the same 240 deletion proposals. Complete-trace reuse saves 48 executions beyond exact-set caching without introducing a wrong label; opening-retrieval matching is cheaper but unsound.

Labeling rule	Fresh exec.	Reused labels	Wrong labels
Per-candidate execution	240	0	0
Exact-set cache	176	64	0
Complete trace	**128**	**112**	**0**
Opening retrieval only	86	154	19

controller makes no model calls. At 30 executions, certification increases from 5/12 to 12/12 and 11/12 under the two schedules, respectively.

A fixed-sequence audit further tests soundness. Among 240 candidates, complete-trace reuse performs 128 fresh executions and reuses 112 labels with no disagreement against per-candidate execution; matching only the first retrieval reduces executions further but produces 19 wrong labels. On two STATE-Bench-derived tool tasks, 34 of 90 candidate pairs with the same first retrieval diverge later, while complete-trace reuse obtains all 26 candidate labels from 12 executions without disagreement. These checks isolate why the complete trace, rather than only the opening context, is the correct reuse unit.

Store size changes retrieval work, not the class structure. The canonical search count depends on retrieval width and repair size once the ranking is fixed, rather than on the number of records stored behind that ranking. We test this directly with nested 1K, 10K, and 100K stores. For minimum repair sizes one, two, and three, the search visits 2, 6, and 16 candidates at every store size; only index maintenance and retrieval time increase. At 100K records the corresponding rebuild-and-retrieve times are 0.48, 1.50, and 3.89 seconds. This experiment separates the combinatorial cost addressed by retrieval equivalence from the engineering cost of maintaining a large store; the full construction and retrieval-width sweep are in Appendix H.

Scaling with reads, width, and repair size. Figure 3 varies $T \in \{1, 3, 6, 10\}$, $q \in \{1, 2, 4\}$, and $k \in \{1, 2, 3\}$ over 108 configurations. Full-trace reuse reduces fresh executions in all 108 configurations and wall-clock time in 107. At $q = 4, k = 3, T = 10$, mean executions fall from 733.00 to 302.33 and end-to-end time from 139.65 ms to 80.25 ms. Appendix D reports component-wise timings and memory-store scaling to 100K records.

6 Related Work

Persistent agent memory and memory poisoning. Persistent-memory agents reuse stored experience to shape later decisions, as in Generative Agents and Reflexion (Park et al., 2023; Shinn et al., 2023). This reuse also creates a delayed attack surface. AgentPoison directly poisons long-term memory or knowledge bases so that triggered queries retrieve malicious demonstrations, while MINJA shows that an attacker can inject harmful memory through query-only interaction without direct access to the store (Chen et al., 2024; Dong et al., 2025). These works establish how memory can cause failures; WitRepair starts after a concrete failure has been observed and asks which stored records must be removed to repair it.

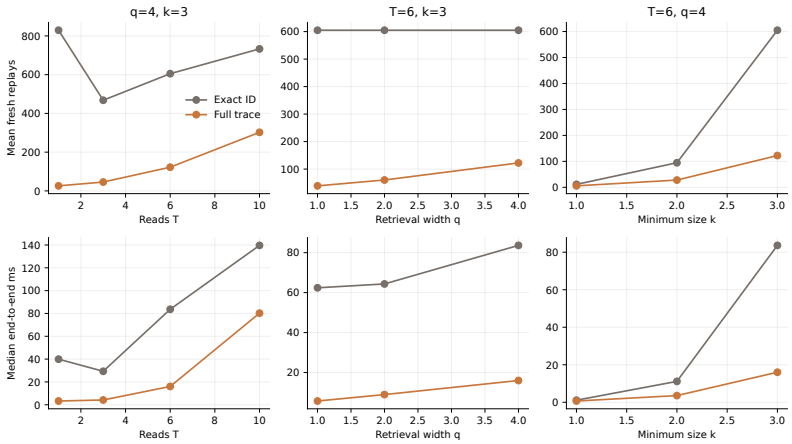


Figure 3: **Multi-turn scaling.** Columns vary the number of reads T, retrieval width q, and minimum repair size k; the top row reports fresh executions and the bottom row end-to-end time.

Memory defenses, auditing, and repair. A-MemGuard compares reasoning paths induced by related memories and stores corrective lessons, while MIND learns intent-aware compressed representations for filtering malicious memory (Wei et al., 2025; Liu et al., 2026). MemAudit addresses the post-hoc attribution problem by combining counterfactual influence with structural anomaly signals to identify memories responsible for harmful behavior (Tan et al., 2026). MEMOREPAIR studies a complementary provenance problem: when source artifacts are invalidated, derived descendants are withdrawn and rebuilt under a barrier-first cascade-repair contract (Zhao et al., 2026). We also adapt the lightweight sequential monitor of Chen et al. (2026) as a runtime comparator; its original setting is long-range decomposition-attack monitoring rather than persistent-memory repair. WITREPAIR searches for a minimum-cardinality deletion within a declared actionable pool and certifies it against a fixed replay map. Its retrieval-equivalence classes allow one execution to resolve several candidate deletions.

Minimal failure sets and diagnosis. Delta debugging searches for smaller failure-inducing inputs by iterative reduction, and model-based diagnosis characterizes explanations through conflicts and corrective sets (Zeller & Hildebrandt, 2002; Reiter, 1987). These lines of work motivate searching over subsets, but they do not exploit the retrieval semantics of persistent-memory agents. WITREPAIR uses the fact that many deletion sets induce the same retrieved context or the same complete execution trace. This structure changes the unit of evidence from an individual subset to an equivalence class and makes one replay informative about many candidate repairs.

Stateful agent evaluation. ToolSandbox and τ-bench evaluate tool-using agents through stateful interaction and task outcomes rather than isolated text responses; STATE-Bench extends this style of evaluation to realistic enterprise workflows and explicitly supports memory-oriented agent evaluation (Lu et al., 2025; Yao et al., 2025; Microsoft, 2026). Our controlled tool experiments follow the same principle: a repair is judged by the resulting task state, while retrieval traces determine when the outcome of one replay can be reused for another candidate.

7 CONCLUSION

WITREPAIR searches for minimum memory repairs by grouping deletions that induce the same retrieval or deterministic execution trace. A representative replay labels its class, and failed-class coverage establishes that no smaller repair exists within the actionable pool. The experiments identify distinct computational benefits: canonical enumeration reduces candidate processing, while complete-trace reuse reduces full executions. The main agent comparison reports improved certified coverage and lower response cost for the combined search policy.

Reproducibility Statement

Appendix A gives the structural proofs, and Appendix B specifies the budgeted search. Appendix C defines the evaluation cohorts, metrics, and cache semantics. The controlled studies and timing aggregation are described in Appendix D.

AI Use Statement

Generative AI tools were used to assist with manuscript writing, experimental design, and analysis of experimental results.

References

Yueh-Han Chen, Nitish Joshi, Yulin Chen, Maksym Andriushchenko, Rico Angell, and He He. Monitoring decomposition attacks with lightweight sequential monitors. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2985777c8f2152664946ed30deb77291-Abstract-Conference.html.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen J. Xiang. Memory injection attacks on LLM agents via query-only interaction. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1554. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/42a97bbd9844d2bf68596730af80bcdf-Abstract-Conference.html.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *arXiv preprint arXiv:2401.08281*, 2024. URL https://arxiv.org/abs/2401.08281.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Dongyi Liu, Haixing He, Xiaobao Wu, and Jia Li. MIND: Lightweight and effective memory injection defense for LLM agents via intent-aware information bottleneck. *arXiv preprint arXiv:2607.28103*, 2026. URL https://arxiv.org/abs/2607.28103.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Microsoft. STATE-Bench: Benchmark AI agents on enterprise workflows. Software repository, 2026. URL https://github.com/microsoft/STATE-Bench.

OpenAI. Model guidance: GPT-5.6. OpenAI API documentation, 2026a. URL https://developers.openai.com/api/docs/guides/latest-model?model=gpt-5.6.

OpenAI. GPT-5.6 Luna Model. OpenAI API documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi: 10.1016/0004-3702(87)90062-2.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Zhewen Tan, Yilun Yao, Huiyan Jin, Wenhan Yu, Guoan Wang, Mengyuan Fan, Liang Lu, Feng Liu, Xiangzheng Zhang, Duohe Ma, Tong Yang, and Lin Sun. MemAudit: Post-hoc auditing of poisoned agent memory via causal attribution and structural anomaly detection. *arXiv preprint arXiv:2605.23723*, 2026. URL https://arxiv.org/abs/2605.23723.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-MemGuard: A proactive defense framework for LLM-based agent memory. *arXiv preprint arXiv:2510.02373*, 2025. URL https://arxiv.org/abs/2510.02373.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002. doi: 10.1109/32.988498.

Yang Zhao, Chengxiao Dai, Mengying Kou, and Yue Xiu. MEMOREPAIR: Barrier-first cascade repair in agentic memory. *arXiv preprint arXiv:2605.07242*, 2026. URL https://arxiv.org/abs/2605.07242.

A Proofs and Structural Details

A.1 Canonical representative and class interval

Fix a legal deletion set R and write its ordered return list as $S = (u_{(i_1)}, \ldots, u_{(i_q)})$ with frontier $\ell = i_q$. Every position $j \le \ell$ is either returned in S or deleted. Hence

$$C_x(R) \subseteq R, \qquad |C_x(R)| = \ell - q.$$

Deleting only $C_x(R)$ leaves exactly the q records of S alive in the prefix through rank ℓ, so the retrieval result is still S. Conversely, any R' returning S must delete every skipped prefix record; otherwise a higher-ranked record would enter the top-q list. Thus $C_x(R) \subseteq R'$. If R' also preserves every element of S, the first q surviving records are exactly S, proving Eq. 5. Therefore $C_x(R)$ is both the unique inclusion-minimum and the unique cardinality-minimum member of the class.

Let $R^\star$ be a minimum-cardinality feasible repair. Canonicalization preserves its execution label and produces $C_x(R^\star) \subseteq R^\star$. Minimum cardinality forces $|C_x(R^\star)| = |R^\star|$, so $C_x(R^\star) = R^\star$. This proves Theorem 1 and the completeness of canonical search.

A.2 Tight single-read query count

For a canonical representative of size d, the q-th survivor appears at rank $q + d$. Among the preceding $q + d - 1$ records, exactly d are deleted, so the number of size-d classes is $\binom{q+d-1}{d}$. Summing over $d \le k$ gives

$$\sum_{d=0}^{k} \binom{q+d-1}{d} = \binom{q+k}{k}.$$

Consider a deterministic algorithm that must be correct for every assignment of binary labels to these classes, with no inference rule relating different class labels. An oracle can answer failure to every query. Until all $Q_{\mathrm{can}}(q,k)$ classes have been observed, an unqueried class may still be feasible, so the algorithm cannot certify infeasibility through size k. This proves the worst-case lower bound; size-ordered canonical search meets it.

The same bound can be attained by a labeling with optimum k: answer failure on all smaller classes and on every queried size-k class except the last such class queried, which succeeds. Finding that success requires inspecting all size-k classes, and certifying it requires excluding every smaller class. This is a worst-case statement about each deterministic search algorithm, with an adversarial labeling chosen for that algorithm.

Proof of Proposition 1. Every legal repair smaller than k has a canonical representative smaller than k. Failure labels for all such classes therefore exclude every smaller repair. Together with the success label of $R^\star$, they establish minimum cardinality. Conversely, an unobserved smaller class may be assigned success while preserving every observed label in the unrestricted black-box model. Its exclusion therefore requires a failure label. The number of smaller classes is $\binom{q+k-1}{k-1}$, giving Eq. 10 after including the successful class. With $A \subset M$, the exact count is one plus the number of reachable classes below k, and the displayed expression remains an upper bound.

Analytic count checks. Enumerating all deletion sets through size k for $q = 1, \ldots, 8$, $k = 1, \ldots, 4$, and $n = q + k$ gives 32 parameter combinations. Grouping by the ordered survivor list recovers the stated class counts and unique minimum representatives in every combination. Table 5 shows selected values.

A.3 Complete-trace class characterization

We prove Eq. 11 by induction over reads. Suppose R' contains all records in D_τ and none in S_τ. At the first read, every record skipped in the recorded execution is still deleted and every returned record is still present, so deletion-stable retrieval returns the same ordered list. The same initial state, context, and deterministic transition produce the same action and tool result, hence the same second query. Repeating the argument preserves every subsequent read and the terminal label. This proves sufficiency.

Table 5: Search queries and certificate labels in the single-read model. The last column assumes the success and failed incident are already known.

q	k	Search Q	Additional search $Q-1$	Labels C	Additional $C-2$
2	1	3	2	2	0
2	2	6	5	4	2
2	3	10	9	7	5
4	2	15	14	6	4
4	3	35	34	16	14
8	3	165	164	46	44

Necessity follows read by read. Any candidate that reproduces the complete trace reaches the same query and ranking at each read. Returning the same list requires deleting every skipped prefix record and preserving every returned record. Taking unions over reads yields $D_\tau \subseteq R'$ and $R' \cap S_\tau = \varnothing$. The candidate that originally produced the trace witnesses $D_\tau \cap S_\tau = \varnothing$, so $R' = D_\tau$ is legal and satisfies all guards. Every class member contains it, establishing the unique minimum representative.

Why opening-only reuse is insufficient. A two-read example shows why the complete trace is necessary. Let $U = \{a, b, c\}$ and $q = 1$. Suppose the first ranking is (a, b, c) and, after reading a, the second ranking is (c, b, a). Both $R = \varnothing$ and $R' = \{c\}$ retrieve a on the first read, but the second read returns c under R and b under R'. A monitor that marks an execution harmful exactly when the second read returns c therefore assigns different outcomes to the two candidates. The complete-trace guard for the original execution requires $c \notin R'$ and correctly blocks this reuse. Equality of the opening retrieval is thus insufficient to certify an interactive replay.

A.4 Multi-turn trace-count bound

Expand execution into an adaptive membership-test tree. The first time a record is encountered along a path, its deletion state branches to deleted or surviving; repeated encounters reuse the previous state. A path contains at most k distinct deleted records and at most qT distinct returned survivors. Let $F(a, b)$ be the maximum number of leaves when a path contains at most a survivor branches and b deletion branches. With $F(a, 0) = F(0, b) = 1$ and

$$F(a, b) \le F(a-1, b) + F(a, b-1),$$

induction gives $F(a, b) \le \binom{a+b}{b}$. Setting $a = qT$ and $b = k$ proves Eq. 12. Tightness is achieved by a construction that exposes qT distinct survivors across T reads while preserving a fixed base order, reducing the trace count to the single-read count with width qT.

A.5 Finite actionable sets and short returns

When $A \subset M$, certification covers only candidates in 2^A. Every reachable class's canonical representative remains legal because it is a subset of the candidate that witnessed the class. Thus the certificate is exact over the declared actionable set A. If fewer than q records remain active, retrieval returns the full active list; the implementation then matches the full activity pattern or the exact candidate, rather than applying the full-width class formula.

A.6 Response caching and complete-trace reuse

Proposition 2 (Fresh responses under matched execution conditions)**.** *Compare two procedures over the same candidate sequence and stopping position, with identical initial evidence, reset states, deterministic transitions, and a fixed model-response map. Both use identical complete, non-evicting full-request caches. Every response in an initially reusable trace is present in both caches. One procedure additionally reuses completed traces when their guards hold. Both procedures incur the same sequence of full-request cache misses.*

Proof. Induct over candidate visits. Without a trace hit, both procedures execute the same candidate, issue the same requests, and update their caches identically. With a trace hit, the guard guarantees

reproduction of a completed trace. Every request in it is already cached, so executing the candidate produces no miss, while trace reuse skips execution. The caches agree after either case. □

Complete-trace reuse can save retrieval, tools, serialization, and cache lookups under these conditions. Fresh-response differences between complete policies depend on candidate order, stopping position, initial evidence, or caching conditions.

B BUDGETED SEARCH PROCEDURE

The implementation maintains an evidence store containing each completed execution, its label, and its class or complete-trace guard. Cache hits do not consume the fresh-response budget.

Algorithm A1. Budgeted WITREPAIR search and certification

1. Input task x, store M, actionable set A, retrieval/execution configuration, and fresh-response budget B.
2. Store the observed incident as failure evidence for the empty deletion set; initialize the evidence store E and fresh-response count $b = 0$.
3. Visit candidates in increasing cardinality. Single-read search enumerates canonical representatives; multi-turn search enumerates legal deletion sets and checks stored complete-trace guards before replay.
4. If an exact-candidate cache entry or a stored trace guard covers the candidate, reuse its terminal label. Otherwise restart from the fixed task state and execute the candidate. Before each model response, first check the full-request cache; on a miss, require $b < B$, generate the response, and increment b.
5. Only a completed execution with safety and validity checks produces a reusable class label. Failed labels update coverage of smaller candidates.
6. On the first successful candidate whose smaller sizes are completely covered by failed evidence, return the candidate, its certificate, and cost b. If a new response is required when $b = B$, return the current size interval and cost. If all legal candidates fail, return certified infeasibility in A.

C MAIN EXPERIMENTAL PROTOCOL

C.1 TASK COPIES AND INCIDENT CONSTRUCTION

The formal evaluation set contains 200 StrategyQA questions, disjoint from development examples and prompt demonstrations. Each task i defines a clean query x_i, a triggered query $\tilde{x}_i$, a clean memory state M_i^{clean}, and an independent poisoned state M_i^{poison}. An incident requires the clean-memory run to be correct and the triggered poisoned-memory run to exhibit the predeclared target behavior. All methods use the same incident list for a given model.

Each method starts from an independent copy of the same poisoned state. Deletion-based methods operate on a predeclared pool of 12 actionable records. Each candidate is applied before the task starts and remains active for the entire replay. The response budget is 30 fresh model generations per incident. If no feasible repair is returned within budget, the frozen poisoned state is retained for downstream evaluation.

For Table 1, each final memory state is evaluated with one clean and one triggered query from independent task resets. Runtime auditing or monitoring baselines are reinitialized for each evaluation query. For Table 2, each incident receives three fresh reruns of the original task and three clean follow-up tasks that were specified before repair and never used to choose a deletion set.

C.2 COMMON PROTOCOL AND ADAPTATION OF MIND

All methods in Table 1 share the same task IDs, initial memory copies, trigger set, retrieval backend, and evaluation queries. Each store contains 64 records. Twelve record IDs form the actionable pool; it contains the six planted records and six benign records, with IDs and attack labels hidden from the repair policy. Retrieval uses exact cosine similarity of frozen MiniLM embeddings, width four, and

ID-based tie breaking. Deletion preserves the relative ranking of surviving records. The clean and triggered evaluation queries are independent of the initial incident replay.

MIND retains its intent–behavior information bottleneck and detector architecture. A frozen Llama-3.1-8B-Instruct extracts representations for all backbones, and detector training uses QA and EHR trajectories disjoint from the evaluation tasks. The threshold is selected on the development set and frozen before testing. At evaluation time the detector filters the common top-4 list without replenishing rejected records. Its memory-write guard is inactive because the repair replay uses a read-only store. Runtime baselines retain their online role; the post-hoc methods act after the shared incident.

The original MIND study uses BGE-large-en retrieval with width five and averages its main results over three seeds and two attack settings (Liu et al., 2026). Here each percentage is derived from integer outcomes on the fixed 200-task protocol. Our ASR-r and ASR-a follow the explicit retrieval-event and conditional-action definitions below. These event definitions also differ from the final-output and reasoning-trajectory events described in MIND's evaluation section. Table 1 therefore compares methods under the common protocol specified here.

Table 6: Protocol differences relevant to the MIND baseline.

Item	MIND publication	This evaluation
Retrieval	BGE-large-en, top-5	MiniLM, exact cosine, top-4
Attacks	AgentPoison and MINJA	AgentPoison
Aggregation	Three-seed averages	Fixed 200-task counts
Intervention	Runtime memory defense	Runtime defense or post-incident deletion
Events	Output and reasoning attack events	Poisoned retrieval and conditional target action

Table 7: Integer outcomes underlying Table 1. C is correct clean answers, P is poisoned-retrieval events, and J is target-action events within P. ACC and ASR-r divide by 200; ASR-a divides J by P.

Model	Method	C	P	J
Llama	Poisoned store	116	94	70
Llama	Full reset	107	0	0
Llama	LLM Auditor	111	68	28
Llama	A-MemGuard	119	66	39
Llama	Sequential Monitor	114	90	62
Llama	MIND	123	60	28
Llama	MemAudit	121	59	27
Llama	WITREPAIR	122	55	22
Qwen	Poisoned store	136	96	77
Qwen	Full reset	115	0	0
Qwen	LLM Auditor	129	38	8
Qwen	A-MemGuard	138	69	43
Qwen	Sequential Monitor	133	92	65
Qwen	MIND	135	34	6
Qwen	MemAudit	143	48	18
Qwen	WITREPAIR	144	43	13
GPT	Poisoned store	142	76	35
GPT	Full reset	138	0	0
GPT	LLM Auditor	140	36	7
GPT	A-MemGuard	144	33	7
GPT	Sequential Monitor	139	40	11
GPT	MIND	143	20	3
GPT	MemAudit	146	51	5
GPT	WITREPAIR	146	49	3

C.3 ATTACK AND REPAIR METRICS

Let c_i indicate whether the clean evaluation query is answered correctly, P_i the injected poison examples, L_i the ordered retrieval result for the triggered query, and a_i whether the final output takes the predeclared target action. Following AgentPoison, define

$$r_i = \mathbf{1}[L_i \neq \varnothing \ \wedge \ \mathrm{set}(L_i) \subseteq P_i], \qquad \mathrm{ACC} = \frac{1}{200}\sum_i c_i, \qquad \text{ASR-r} = \frac{1}{200}\sum_i r_i,$$

$$\text{ASR-a} = \frac{\sum_i r_i a_i}{\sum_i r_i},$$

with ASR-a reported as n/a when $\sum_i r_i = 0$. Percentages multiply these proportions by 100. Table 2 measures residual target output over all $3N$ fresh incident reruns, without conditioning on r_i.

For the incident-level evaluation, safe completion requires both safety and correctness on a fresh original-task rerun. Residual target output counts a rerun whose final output contains the predeclared target. Repair rate, minimum-hit rate, and certificate rate use all N incidents as denominator. Minimum hits are checked against an independent exact optimum over the 12-record actionable pool. Mean deletion size is reported on a fixed common-success cohort C of 40 Llama, 40 Qwen, and 30 GPT incidents shared by all five deletion methods. Cohort selection uses success indicators, with task IDs breaking ties, and never uses deletion size or follow-up outcomes. Fresh-response cost includes every uncached response used to acquire, refine, compare, or certify a repair; final evaluation queries are counted separately.

C.4 MONITOR SEMANTICS ACROSS EVIDENCE SOURCES

Table 8 defines the environment-specific safety and validity predicates. The routing study checks action format; the tool studies check executed task state.

Table 8: Monitor semantics used by the three experimental evidence sources.

Setting	Harm / failure evidence	Validity used in reporting
Frozen-response routing	Declared risky action in parsed model output	V_{fmt}: legal, parseable action; no executed-task claim
STATE-Bench-derived tasks	Memory-induced failure plus final-state audit	V_{task}: task completion assertions on the reset database
SQLite controller	$H = 1$ iff a retrieved faulty instruction executes the explicit leave-active action	$V_{\mathrm{task}} = 1$ iff all three final booking flags are zero

For the SQLite controller, a successful repair requires $H = 0$ and $V_{\mathrm{task}} = 1$. Each study reports outcomes under its own predicate.

C.5 BASELINES AND CACHE SEMANTICS

LLM Auditor. A language-model auditor removes manipulative or logically inconsistent retrieved content before it reaches the agent, following the audit baseline used with A-MemGuard (Wei et al., 2025).

A-MemGuard. The defense evaluates reasoning consistency across memory-conditioned paths and stores corrective experience for later action selection (Wei et al., 2025).

MIND. MIND compresses intent–behavior features through an information bottleneck and classifies memory as benign or malicious (Liu et al., 2026).

Sequential Monitor. We adapt the lightweight sequential monitor of Chen et al. (2026), originally proposed for detecting long-range decomposition attacks, as a generic runtime comparator that accumulates evidence across requests. It intervenes during task execution.

MemAudit-D. We use MemAudit's post-hoc memory attribution and structural signals to select records for deletion; it returns a repair set but not a minimum certificate (Tan et al., 2026).

Delete+reinsert. The method deletes failure-associated retrieved records until a repair is found, then restores records in a fixed order with a fresh replay after each change.

Reinsert+certify. After reinsertion, the method continues exact search over smaller candidates until it has enough evidence to certify or exhausts budget.

Exact+cache. Exact search orders candidates by $(|R|, \mathrm{IDs}(R))$ and reuses full model requests. WITREPAIR orders candidates by $(|R|, \sum_{u \in R} \mathrm{rank}_0(u), \mathrm{IDs}(R))$, with ranks taken from the initial incident and stable ID tie breaking. Both use the same initial evidence and complete, non-evicting request cache. All responses from the initially reusable incident trace are present.

We distinguish three caches. An *exact-set cache* stores the terminal result of a particular deletion set. A *full-request cache* stores model responses keyed by model identifier, messages, tools, and decoding configuration. A *trace-evidence cache* stores (D_τ, S_τ) and the terminal label, allowing reuse across different deletion sets satisfying Eq. 8.

C.6 COUNTS UNDERLYING TABLE 2

Table 9: Counts for WITREPAIR in the three main model groups.

Model	Safe reruns	Residual target	Repair/min-hit/cert.	Follow-up success
Llama-3.1-8B-Instruct	144/180	10/180	56/60	135/180
Qwen3-8B	195/234	10/234	75/78	180/234
GPT-5.6	103/117	2/117	38/39	106/117

D MECHANISM EXPERIMENTS

D.1 CANONICAL ENUMERATION AND SEARCH ORDER

The frozen-response study uses five checkpoints (Qwen3-8B-AWQ, Gemma2-2B, Qwen2.5-1.5B, Qwen3-0.6B, and TinyLlama-1.1B), retrieval widths $q \in \{4, 6, 8\}$, and 12 eight-record scenarios. Of 180 scenario–model–width grids, 97 contain a harmful, format-valid incident with a clean reference that passes the same checks.

Table 10: Frozen-response search and certification over 97 incidents.

Method	Repaired	Min. hit	Certified	Mean model calls
Iterated deletion	96	84	59	8.09
Delete+reinsert	96	94	62	8.84
Delete+exact certification	97	97	97	11.57
WITREPAIR canonical search	97	97	97	8.03

Direct canonical search reduces mean calls by 30.6% relative to repair followed by exact certification. When visit order and prompt caching are fixed, general enumeration performs 10.61 candidate evaluations with 8.03 model calls, while canonical enumeration performs 8.03 candidate evaluations with the same 8.03 model calls. This isolates canonicalization as removal of redundant candidate processing, while search order changes when new class observations are acquired.

D.2 TRACE-REUSE TIMING DECOMPOSITION

The SQL cross experiment uses minimum repair sizes $k \in \{1, 2, 3\}$ and four layout seeds. Every task creates a fresh database, processes flight, hotel, and car stages, and uses three state-dependent

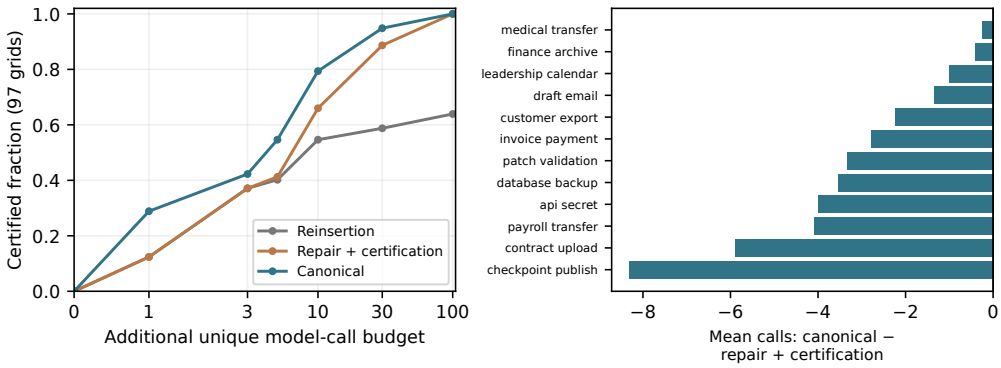


Figure 4: Frozen-response certification progress and paired model-call differences.

top-2 reads under a scripted controller. Each configuration is timed five times. Table 3 takes the median of the five runs for each configuration, then the median across the twelve configurations. The component times below are arithmetic means over the same configuration–repeat runs; their total need not equal the median in Table 3.

Table 11: Mean SQL timing decomposition in milliseconds.

Search/reuse	Exec.	Match	Generate	Cover	Other	Total
Repair-first / exact candidate	3.637	0.000	0.025	0.014	0.046	3.722
Repair-first / full trace	0.834	0.035	0.019	0.012	0.033	0.933
Direct / exact candidate	6.021	0.000	0.050	0.017	0.071	6.159
Direct / full trace	1.001	0.055	0.036	0.013	0.041	1.146

Trace matching adds small bookkeeping cost but eliminates enough full executions to reduce total time substantially in this controlled setting.

D.3 Scaling with T, q, and k

The scaling study uses 20 records, $T \in \{1, 3, 6, 10\}$ reads, $q \in \{1, 2, 4\}$, minimum repair sizes $k \in \{1, 2, 3\}$, and three layout seeds, for 108 configurations and 432 policy runs. For $n = 20$ and $k = 3$, there are only $\sum_{d=0}^{3} \binom{20}{d} = 1{,}351$ legal deletion sets. We therefore use the finite-store bound $\min\{\binom{qT+k}{k}, \sum_{d=0}^{k} \binom{n}{d}\}$. At $q = 4, k = 3$, this bound and the direct-search costs are:

Table 12: Read-depth scaling for $q = 4, k = 3$.

T	Class bound	Candidate exec.	Trace exec.	Candidate ms	Trace ms
1	35	829.33	26.00	39.93	3.28
3	455	468.00	45.67	29.34	4.17
6	1,351	605.00	122.33	83.65	16.01
10	1,351	733.00	302.33	139.65	80.25

The theoretical count is an upper bound on distinct trace classes up to size k; actual execution cost depends on the path structure, visited candidates, and the point at which search terminates.

D.4 Memory-store scale

Appendix H gives the vector-store construction and the nested 1K–100K evaluation. Table 18 reports candidate counts and retrieval/index time for the three optimum sizes.

D.5 FIXED CANDIDATE-SEQUENCE AUDIT

A fixed sequence contains 240 deletion candidates, including 64 exact duplicates. Per-candidate execution requires 240 fresh executions; exact-set caching requires 176; complete-trace reuse requires 128 and reuses 112 labels with zero disagreement. First-retrieval matching needs only 86 executions but introduces 19 incorrect labels because later reads diverge. A separate SQL ledger yields 418 label disagreements and 2,120 trajectory disagreements under first-read matching, versus zero under complete-trace reuse.

E REPAIR CASES AND FOLLOW-UP UTILITY

E.1 JOINT REPAIR AND CERTIFICATE

In the three-stage cancellation example, the successful deletion set is $\{u_2, u_5\}$. The three returns are (u_0, u_4), (u_6, u_4), and (u_4, u_{10}), giving

$$D_\tau = \{u_2, u_5\}, \qquad S_\tau = \{u_0, u_4, u_6, u_{10}\}.$$

The certificate records the failure of the empty set and all single-record candidates, together with their retrieval conditions and final task checks. The failed smaller candidates and the successful size-two trace establish minimum cardinality two.

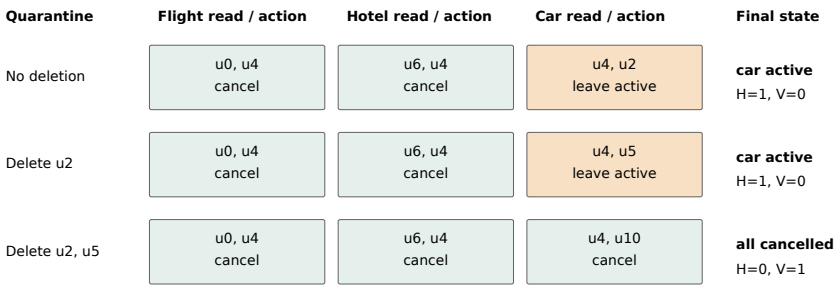


Figure 5: Controlled joint-repair case. Deleting one harmful record exposes a substitute; deleting both changes the final retrieval and restores the target state.

E.2 WHICH MINIMUM REPAIR IS SELECTED MATTERS

We construct 48 SQL incidents from $k \in \{1, 2, 3\}$ and 16 layouts. Each stage contains two records that jointly induce failure and one correct instruction; deleting either one of the pair exposes the correct record. The resulting 2^k solutions all have minimum size k but retain different memories. Three predeclared follow-up tasks per incident depend on these retained instructions.

Table 13: Follow-up utility for equally small certified repairs.

Search method (full-trace reuse)	Certified incidents	Mean deletions	Follow-up success
Repair-first + certification	48/48	2.0	74/144
Direct exact search	48/48	2.0	64/144

The two searches disagree on 82 follow-up tasks: 46 are solved only by repair-first search and 36 only by direct search. A bootstrap over incident blocks stratified by optimum size gives a 95% interval of $[-3.47, 17.36]$ percentage points for repair-first minus direct search. The mean difference is 6.94 percentage points, with uncertainty spanning both directions. The 82 discordant outcomes show how equally small repairs retain different task-relevant records.

E.3 STATE-BENCH-DERIVED TOOL EXECUTION

We additionally evaluate a Qwen3-8B-AWQ travel agent on a local protocol derived from STATE-Bench (Microsoft, 2026). The agent interacts with resettable task databases and retrieves top-3 memories per turn from 100 training-trajectory records embedded with MiniLM and indexed by exact cosine search in FAISS (Wang et al., 2020; Douze et al., 2024). A predeclared 12-record opening pool supplies candidate interventions; every candidate is re-run from a fresh database state and is scored by executed final-state assertions.

We use disjoint 12-task development and confirmatory cohorts. In the confirmatory cohort, four tasks are valid without memory and six with the full store; exactly one no-memory-competent task exhibits a memory-induced failure. A one-record deletion repairs that incident. Because the empty deletion is already observed to fail, the feasible singleton closes a minimum-size-one certificate. Figure 6 shows the complete state transition and proof frontier rather than only the final text response.

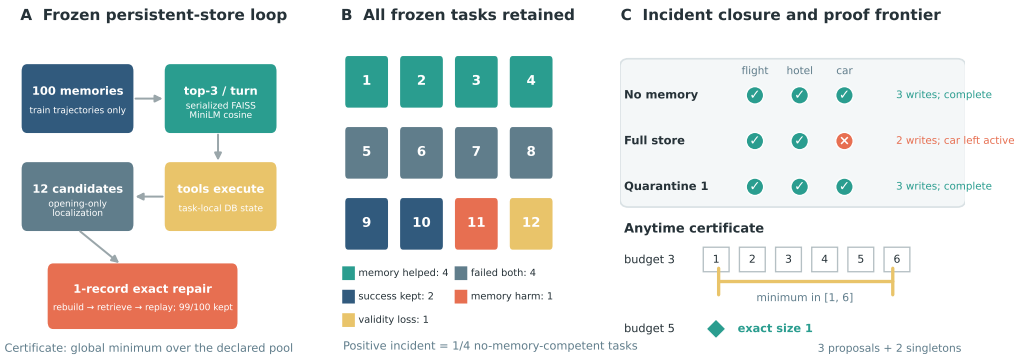


Figure 6: **STATE-Bench-derived held-out tool execution.** The persistent-store loop re-retrieves after intervention; the positive confirmation incident leaves the car reservation active with the full store and restores the required final state after a singleton deletion. The failed empty set and feasible singleton certify minimum cardinality one over the declared repair namespace.

Trace-guard audit. For two held-out tool tasks, the empty deletion and all 12 singletons produce 26 candidate executions occupying eight opening-retrieval classes. Among 90 pairs with the same opening retrieval, 34 diverge at a later read. In contrast, all 112 cross-candidate pairs satisfying a complete-trace guard reproduce both the recorded trace and terminal label. A sequential trace memoizer labels all 26 candidates with 12 fresh executions, saving 14 executions (53.8%) without a label disagreement. This audit directly tests the soundness distinction illustrated by the opening-only counterexample above.

F EXTENSIONS

F.1 NONNEGATIVE DELETION COSTS

If record u_i has nonnegative intervention cost w_i, the objective becomes

$$\min_{R \subseteq A} \sum_{u_i \in R} w_i \qquad \text{s.t.} \qquad \Phi_x(R) = 1.$$

Because $C_x(R) \subseteq R$, canonicalization never increases this cost. Therefore at least one cost-optimal solution is canonical. Best-first search over accumulated weight can replace cardinality order, while the certificate covers all candidates with lower total cost.

F.2 STOCHASTIC REPLAY

For stochastic execution, let $H(R), V(R) \in [0, 1]$ denote harmfulness and task validity and target $\mu_h(R) \leq \tau_h$, $\mu_v(R) \geq \tau_v$. Suppose at most Q candidates are adaptively tested and each is sampled

at most L times. For $j \in \{h, v\}$, use

$$\epsilon_t = \sqrt{\frac{\log(4QL/\delta)}{2t}}, \qquad I_{j,t} = [\widehat{\mu}_{j,t} - \epsilon_t, \widehat{\mu}_{j,t} + \epsilon_t] \cap [0, 1].$$

Accept when harmfulness is certified below threshold and validity above threshold; reject when either interval certifies a violation; otherwise continue sampling or abstain at budget exhaustion. Assuming independent samples from a fixed outcome distribution for each candidate, Hoeffding's inequality and a union bound over both statistics, at most Q selected candidates, and L sample sizes give simultaneous interval coverage of at least $1 - \delta$ (Hoeffding, 1963). A statistical minimum certificate additionally requires accepting the returned repair and rejecting every smaller legal candidate within this joint event. Unresolved candidates prevent certification.

F.3 General deletion solvers

When the repair oracle has additional monotone structure, failed active-memory cores can be converted into hitting constraints and valid-memory cores into preservation constraints. With binary deletion variables x_i, a harmful witness C yields $\sum_{i \in C} x_i \geq 1$, while a validity core D yields $\sum_{i \in D} x_i \leq |D| - 1$. These constraints can be combined with exact no-good constraints and solved by branch search or MILP. This extension is separate from retrieval-equivalence search; the former exploits logical constraints in the oracle, while WitRepair exploits retrieval-induced equivalence in the execution map.

G Additional Frozen-Model and Layout Audits

These audits vary prompt layout, model checkpoint, and candidate order. They use the deletion objective from Eq. 2 with the routing-specific validity predicate in Table 8.

G.1 Frozen-response corpus

The frozen-response corpus stores a response for each of the 256 active-memory subsets in twelve eight-record scenarios, two prompt layouts, and five local checkpoints. Each retrieval width $q \in \{4, 6, 8\}$ maps a deletion candidate to a returned subset in this shared response corpus; widths do not multiply the stored row count. Prompts request exactly one available tool identifier, expose only opaque memory IDs, and keep planted labels out of the model input. The neutral-newest layout orders dated updates newest first. The dual-reordered layout permutes memory order, reverses tool order, and instructs the model to rely on content rather than position. Greedy generation returns at most twelve new tokens; the experiment records the complete prompt and response so that every policy is evaluated on the same deterministic prompt-to-response map.

Table 14: Frozen-response corpus. Each checkpoint contains 12 scenarios, 256 memory subsets, and two layouts.

Checkpoint	Rows per layout pair	Retrieval widths
Qwen3-8B-AWQ	6,144	4, 6, 8
Gemma2-2B-it	6,144	4, 6, 8
Qwen2.5-1.5B-Instruct	6,144	4, 6, 8
Qwen3-0.6B	6,144	4, 6, 8
TinyLlama-1.1B	6,144	4, 6, 8

For every grid we enumerate feasible masks, minimum repair size, and optimal masks. The layout audit uses the full active list ($q = 8$) over 120 model–scenario–layout cells, separately from the 180-cell width audit. Across the two layouts there are 70 positive grids; the audited optimum sizes are one for 25 grids, two for 30, and three for 15. Twenty-nine model–scenario units remain positive under both layouts. This layout pairing is useful because it changes surface order while preserving the underlying memory contents.

Table 15: Unfiltered positive incidence and singleton invisibility, reported as positive / singleton-invisible grids out of 12 scenarios.

Checkpoint	Neutral-newest	Dual-reordered
Qwen3-8B-AWQ	11 / 7	10 / 7
Gemma2-2B-it	9 / 6	6 / 5
Qwen2.5-1.5B	12 / 9	12 / 6
Qwen3-0.6B	8 / 4	2 / 1
TinyLlama-1.1B	0 / 0	0 / 0
All	40 / 26	30 / 19

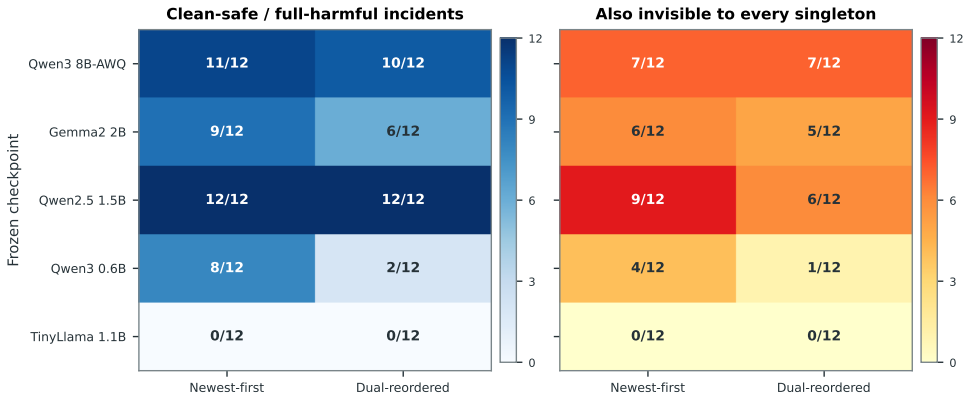


Figure 7: Incidence audit across checkpoints and prompt layouts. Each cell reports results over all twelve scenarios.

G.2 Hosted-model transfer

We also evaluate twelve persistent-store repair cases with the hosted `gpt-5.6-luna` API (OpenAI, 2026b). All 12 cases are repaired and hit a minimum repair, 11/12 complete a certificate within the recorded budget, and the mean additional call count is 7.3. The certificate is conditional on the frozen request/response snapshot. Table 16 reports the aggregate recovery and certification outcomes.

Table 16: Hosted-model transfer on twelve repair cases.

Cases	Repaired	Minimum hit	Certified	Mean additional calls
12	12/12	12/12	11/12	7.3

H Persistent Vector-Store Closure and Store-Size Scaling

The main text separates class-query complexity from store maintenance cost. This section gives the full persistent-vector-store protocol used for that comparison.

Each of twelve task families has an independent namespace containing three planted false writes and five benign records. Records are embedded with a frozen MiniLM encoder, normalized, and indexed for exact cosine retrieval with deterministic tie breaking (Wang et al., 2020; Douze et al., 2024). For each deletion candidate the survivor index is rebuilt, the task is re-retrieved from the modified store, and the frozen agent response is evaluated. Thus candidate labels come from the post-intervention retrieval state rather than from the original context.

The audited optimum sizes are one, two, and three in 46, 36, and 15 incidents. The exact search removes 161 planted false writes and only two benign records over the 97 repaired incidents. The sole iterated-context failure occurs when no subset of the initial top-4 context is sufficient, while a

Table 17: Persistent-store closure over 97 positive incidents. "Overall" is repaired incidents; mean deletions are reported over the corresponding successful cohort.

Policy	Overall	Top-4	Top-6	Top-8	Mean $\lvert R\rvert$
Context one-shot	79/97	18/26	23/33	38/38	1.63
Iterated context	96/97	25/26	33/33	38/38	1.78
Delete initial top-q	64/97	13/26	19/33	32/38	6.25
Store-aware exact	**97/97**	**26/26**	**33/33**	**38/38**	1.68

three-record deletion elsewhere in the actionable namespace repairs the task after re-retrieval. This case exhibits the replacement effect in Figure 1: a deletion changes which memory is retrieved next.

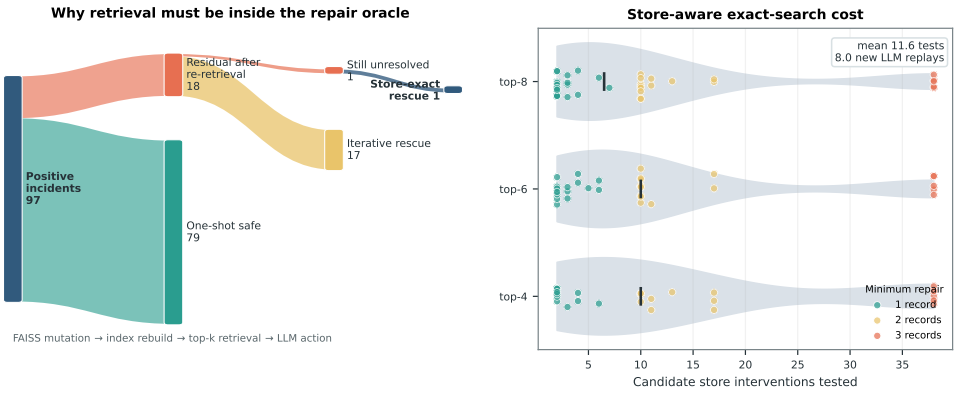


Figure 8: Persistent-store mutation and re-retrieval. Left: escalation after the store changes. Right: the distribution of candidate store interventions.

H.1 Nested 1K–100K stores

We build nested stores of 1K, 10K, and 100K records while preserving the target incident and its retrieval frontier. The three target tasks have known optimum sizes one, two, and three. Candidate counts stay fixed at 2, 6, and 16 across the three store sizes, while index rebuild and retrieval time grow with the number of stored records.

Table 18: Store scaling at fixed retrieval width. Tests are reported for 1K / 10K / 100K records.

Incident	$k^\star$	Candidate tests	Class bound	100K index+retrieve (s)
Invoice payment	1	2 / 2 / 2	5	0.48
Database backup	2	6 / 6 / 6	15	1.50
API credential	3	16 / 16 / 16	35	3.89

A separate breadth grid grows the namespace from 8 to 128 records. All positive grids are repaired exactly; median canonical tests remain between 3.0 and 3.5. At 100K records, varying q changes the number of reachable classes as predicted by the theory: the maximum observed test counts are 2, 16, and 46 for $q = 2, 4, 8$, respectively. These results distinguish namespace growth from retrieval-width growth.

I Matched-Order Budget and Certification Audits

This audit expands the matched-order comparison. Each policy starts with only the observed incident cached. A new exact prompt costs one model call; cache hits are free. We separately count candidate/oracle evaluations because canonicalization can reduce candidate processing even when prompt caching collapses multiple candidates to the same response.

Table 19: Retrospective budget audit over all 97 positive frozen-response incidents. Counts are incidents; call and oracle columns are means over all incidents.

Budget	Policy	Repair	Min. hit	Cert.	Model calls	Candidate evals
10	One-shot context	61	60	51	5.42	5.73
10	Iterated context	77	72	51	5.69	6.25
10	General exact search	77	77	77	5.25	6.86
10	Canonical, matched order	77	77	77	5.25	5.25
30	One-shot context	74	67	54	7.45	7.79
30	Iterated context	91	79	54	7.73	8.34
30	General exact search	92	92	92	7.67	10.25
30	Canonical, matched order	92	92	92	7.67	7.67
Full	One-shot context	79	72	59	7.81	8.21
Full	Iterated context	96	84	59	8.09	8.75
Full	General exact search	97	97	97	8.03	10.61
Full	Canonical, matched order	97	97	97	8.03	8.03

The matched-order rows isolate the representation effect: general and canonical exact search use the same model-call frontier under full prompt caching, while canonicalization removes repeated candidate evaluations. The end-to-end policy comparison additionally changes visit order, which is why direct canonical search uses fewer model calls than repair followed by exact certification in Table 10.

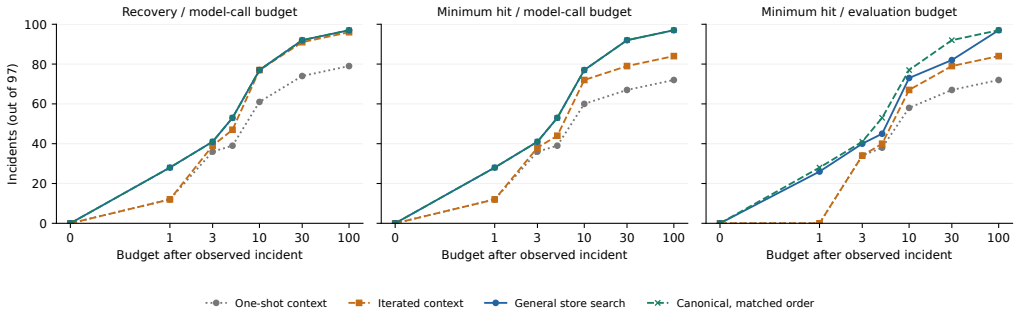


Figure 9: Budget frontiers from the frozen-response audit. Model-call curves can overlap after caching even when candidate-evaluation frontiers differ.

J Executed Stateful-Tool Audit

The SQL experiments use a scripted controller over resettable SQLite state with three reads and three action stages (flight, hotel, and car). Every candidate starts from the same database state, and the monitor checks the terminal booking flags. The tables below report complete-search outcomes and equal-budget certificates. Timing uses the repeated-run cohort and aggregation specified in Appendix D.

The fixed-ledger audit holds the 240-proposal sequence constant. Exact candidate caching reuses 64 duplicate proposals; complete-trace guards reuse an additional 48 labels with zero disagreement. Opening-retrieval-only reuse is an intentionally unsound control: it performs only 86 fresh executions but assigns 19 incorrect labels because later reads diverge. A larger ledger audit produces 418 label disagreements and 2,120 trajectory disagreements under opening-only reuse, versus zero under complete-trace guards.

Table 20: Executed SQL policies at completion, over twelve configurations. Timing matches Table 3; "–" indicates that the reinsertion-only policy has no five-repeat timing cohort.

Policy	Repaired	Certified	Mean fresh exec.	Median ms
Reinsertion	8	4	3.33	–
Repair + certification	12	12	42.42	3.62
Same policy + trace reuse	12	12	9.58	1.03
General exact	12	12	74.25	3.54
General exact + trace reuse	12	12	11.75	0.95

Table 21: Equal-budget SQL audit. Cells are repaired / minimum-hit / certified configurations out of twelve.

Policy	10 replays	30 replays	60 replays
Iterated + reinsertion	8/8/4	8/8/4	8/8/4
Reinsertion + certification	8/8/4	9/9/5	11/11/7
Same policy + trace reuse	9/9/5	12/12/12	12/12/12
General exact	4/4/4	5/5/5	7/7/7
General exact + trace reuse	6/6/6	11/11/11	12/12/12

J.1 Joint repair with an explicit certificate

Figure 5 shows the size-two cancellation case. With no deletion, the final read returns (u_4, u_2) and the car remains active. Deleting u_2 exposes u_5; deleting both returns (u_4, u_{10}) and cancels all bookings. The empty set and all twelve singletons fail, establishing minimum size two within the twelve-record pool.

K Multi-Environment Evaluation and Repair-Size Breakdown

The multi-environment evaluation contains 60 positive incidents from travel cancellation, invoice approval, and account-permission workflows (20 each), using the same three-stage scripted controller interface and a twelve-record actionable namespace per incident. The audited minimum sizes are one for 20 incidents, two for 24, and three for 16. Budgets count fresh trajectories after the initial incident.

Table 22: Multi-environment outcomes at 30 and 60 replays. Cells are repaired / minimum hit / certified.

Policy	30 replays	60 replays
Deletion + reinsertion	48/41/13	53/46/24
Reinsertion + certification	49/45/35	55/52/47
Trace-based repair system	51/49/44	57/56/53

The trace-based system is evaluated at a trajectory budget in this cohort. Tables 22 and 23 report its recovery and certification outcomes.

The common-success subset contains 48 incidents and three follow-up tasks per incident. The trace-based system completes 135/144 follow-up tasks while retaining 96% of benign records with 2.0 mean deletions; reinsertion + certification completes 126/144 with 2.1 mean deletions. This follow-up cohort has 144 trials; the main StrategyQA cohorts use the denominators in Table 2.

Table 23: Repair-size breakdown at the 60-replay budget. Cells are repaired / minimum hit / certified.

Policy	$k^\star = 1$ ($n = 20$)	$k^\star = 2$ ($n = 24$)	$k^\star = 3$ ($n = 16$)
Deletion + reinsertion	20/20/20	21/18/4	12/8/0
Reinsertion + certification	20/20/20	22/21/18	13/11/9
Trace-based repair system	20/20/20	23/23/22	14/13/11