When Does a Memory Descendant Need Repair? Structural Lineage vs. Stale State in Persistent Agent Memory

Anonymous authors
Paper under double-blind review

Abstract

Persistent LLM agents store conclusions that can remain active after their sources are corrected. We show that structural provenance fidelity and repair adequacy are different quantities. We distinguish structural descendants, descendants whose task value changes under an update, and descendants whose stored value is wrong afterward. In a controlled persistent-memory testbed with two LLM readers, material-support provenance covers about two-thirds of structural descendants at shallow depth and one-half at greater depth, while reaching at least 95% of stale stored state. Repair remains near oracle with fewer re-derivations. Matched-random provenance with the same edge budget and structural precision/recall reaches only about two-thirds of stale state. A second attributor improves structural fidelity while reaching the same stale state with more work. A symbolic value-agreement graph, retaining parents aligned with the stored conclusion, reproduces the LLM graph's selectivity. For unate rules and correct stored values, we prove that current-state value-agreement support reaches every update-sensitive descendant. Public multi-hop memory experiments show that persistent intermediates sharply lower post-update accuracy when they become invalid, with little change when they remain valid. These results establish stale-state coverage, repair work, and post-repair behavior as essential measures of provenance quality alongside structural fidelity.

1 Introduction

Persistent LLM agents increasingly reuse materialized state instead of recomputing every conclusion from raw history. Reflections summarize observations, intermediate conclusions support later decisions, and memory systems retain compact artifacts across interactions (Park et al., 2023; Shinn et al., 2023; Packer et al., 2023; Xu et al., 2025). Persistence creates a maintenance problem once upstream information changes: correcting a source does not automatically remove a descendant that was written earlier and is still retrievable.

Stale descendants change subsequent answers. In our controlled lifecycle, removing a stale derived item raises post-audit accuracy from 0.690 to 0.856, a gain of 0.166 (95% CI $[0.111, 0.221]$). Replacing it with unrelated derived text of the same form and length yields 0.847, while prompt-order changes have smaller effects (Appendix Fig. 5). Across longer audit delays, root-only correction accumulates residual probe error while provenance-closure repair stays close to the no-write-back reference (Appendix Fig. 6). These observations connect downstream memory repair (Zhao et al., 2026; Yu et al., 2026) to a concrete behavioral consequence of the stored content.

Which dependencies must provenance retain for repair to succeed? A structural graph records which stored states depended on a source. Repair after a particular correction targets active stored states whose values disagree with the updated task. A descendant may retain its task value after an upstream change. A reader may also have stored a wrong answer that remains wrong, or an answer that becomes correct after the update. Repair need therefore depends jointly on the correction and the materialized value.

We therefore separate three objects for each audited root u at time t (Fig. 1): structural descendants $\mathcal{A}_{\mathrm{struct}}(u, t)$, audit-time update-sensitive descendants $\mathcal{A}_{\mathrm{sens}}(u, t)$, and post-audit stale stored descen-

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.

dants $\mathcal{A}_{\text{stale}}(u, t)$. The first records historical lineage; the second records whether the task-semantic value changes under the current update; the third records whether the value actually stored by the reader is wrong after that update. In the observed graph replays, the last two coincide for DeepSeek and differ through reader errors for Luna, while structural lineage is far larger than either.

The resulting separation is large. As derivation depth increases, LLM material-support attribution covers fewer structural descendants: coverage falls from about 0.68 to 0.50–0.51. Stale-state coverage stays at 0.95–1.00, with at most 2.3% unreached and near-oracle post-repair behavior. Holding the structural edge budget and aggregate precision/recall fixed, matched-random provenance reaches only 0.66–0.68 of stale state. A symbolic but-for graph with perfect edge precision also leaves more stale state unreached. A symbolic *value-agreement* graph, retaining parents aligned with the stored conclusion, reproduces the LLM result. This identifies the support selected by material attribution and motivates the reachability guarantee for unate rules in Proposition 1.

Our contributions are:

- **An event-level separation of lineage from repair need.** We define structural, update-sensitive, and stale descendants against the actual state immediately before each audit, and show that for unate rules with correct stored values, value-agreement support provably reaches every update-sensitive descendant.
- **A controlled separation of graph fidelity and repair adequacy.** Structural coverage can fall sharply while stale-state coverage and post-repair behavior remain near oracle; edge-budget-matched random and symbolic controls separate selective repair value from graph sparsity, and a value-agreement control identifies what the LLM graph selects.
- **A behavioral validation on public multi-hop memory data.** Persistent intermediates cause a large post-update penalty when they become invalid, whereas persistence is much less harmful when the intermediate remains valid.

2 Related Work

Persistent memory and agent repair. Agents retain observations, reflections, and compact memories across interactions (Park et al., 2023; Shinn et al., 2023; Packer et al., 2023; Xu et al., 2025). These stored artifacts become repair targets when their sources change. MEMOREPAIR withdraws affected descendants and republishes validated successors (Zhao et al., 2026); dependency-guided rollback preserves independently supported descendants and replays answer-relevant affected computation (Yu et al., 2026). REVISE follows recorded data and control dependencies while retaining valid progress under online revisions (Qi et al., 2026). GRADE records execution and dependency structure in agent traces (Zhao, 2026), and revocation experiments show how superseded records continue to influence behavior when retrieval fails to enforce invalidation (Shen et al., 2026).

Provenance and semantic dependence. Database provenance characterizes the relationship between inputs and outputs through why-/where-provenance (Buneman et al., 2001) and algebraic annotations (Green et al., 2007). Dependency provenance connects this relationship to program analysis and gives a semantic account of how input changes affect outputs (Cheney et al., 2007). These perspectives distinguish recording a derivation from characterizing its dependence on particular inputs. Our structural graph records the parents required by the task rule; update sensitivity evaluates the rule under one correction, and staleness compares its new value with the answer actually stored by the reader.

Witnesses and sufficient reasons. Witnesses in database provenance identify input subsets sufficient for an output (Buneman et al., 2001; Green et al., 2007). Abductive explanations and sufficient reasons formalize minimal feature assignments that preserve a classifier's decision (Ignatiev et al., 2019; Darwiche and Hirth, 2020). For a correct output of a unate rule, sign normalization makes value-agreement support contain the union of all minimal sufficient reasons (Appendix C). This connects our support graph to established sufficiency concepts. Our contribution is to identify its repair value: it retains paths to update-sensitive state, reproduces LLM attribution's selectivity, and separates stale-state coverage from structural fidelity.

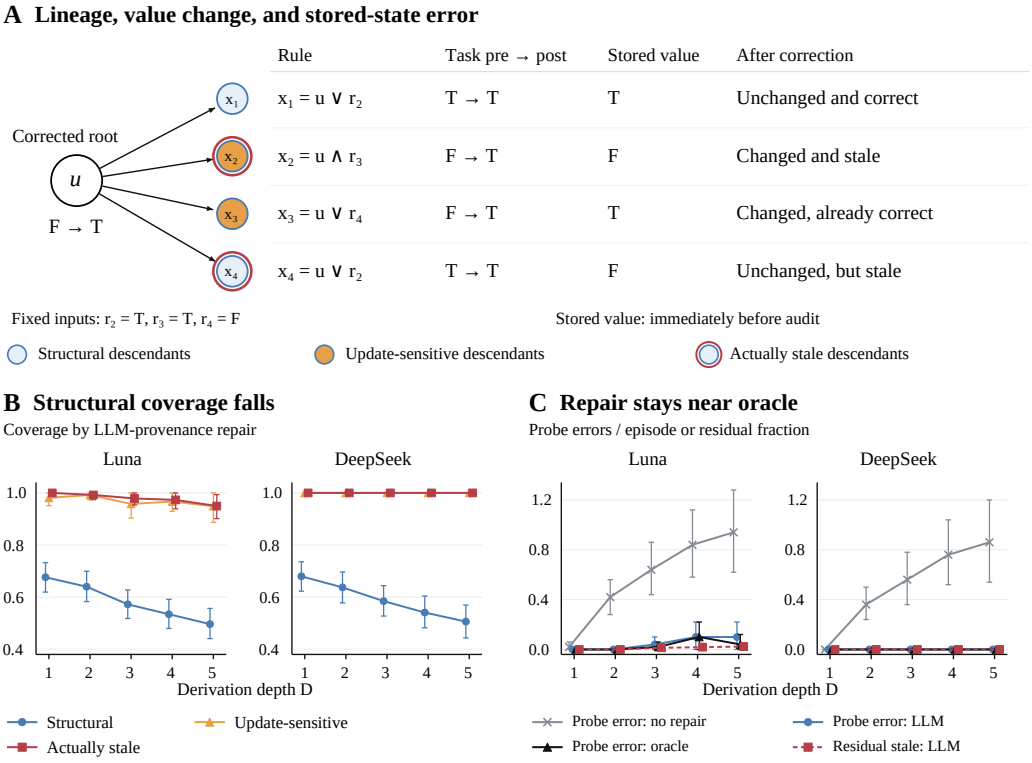

Figure 1: **Structural lineage, update sensitivity, and stored-state staleness are different repair-relevant objects. A:** An audited root can have structural descendants whose task value does not change, descendants whose task value changes, and descendants whose actually stored value is stale after the update; the sets need not be nested. In this example, sensitive-but-fresh (x_3) and insensitive-but-stale (x_4) states arise from reader errors. **B:** Under LLM material-support provenance, structural descendant coverage falls with derivation depth while audit-time sensitive and stale-state coverage remain high. **C:** The lifecycle consequence follows stale-state coverage: residual stale state stays near zero and post-repair probe error remains close to oracle. Error bars are episode-cluster paired bootstrap 95% CIs over 50 paired episodes.

Causality and overdetermination. Structural-model accounts of actual causality use counterfactual interventions and contingencies to handle cases with multiple sufficient causes (Halpern and Pearl, 2005). Database causality similarly distinguishes query lineage from the causes of a particular answer (Meliou et al., 2010). Our but-for control tests a specific local criterion: flipping one parent at formation while holding all other inputs fixed. Sequential corrections expose its consequence for memory repair: a parent that is individually non-decisive at formation can become decisive after other values change.

Maintaining derived state. Truth-maintenance systems track justifications and the assumptions under which beliefs hold (Doyle, 1979; de Kleer, 1986). Incremental view maintenance updates stored query results through counting and delete-and-rederive procedures (Gupta et al., 1993). Adaptive computation propagates input changes through recorded dependencies (Acar et al., 2002), while Adapton combines dependency tracking with demand-driven recomputation and reuse (Hammer et al., 2014). These systems establish dependency-guided maintenance as a computational foundation. We evaluate the provenance supplied to a fixed repair operator, relating structural fidelity to the stale stored answers it reaches, the work it performs, and the answers produced afterward.

3 THREE REPAIR-RELEVANT OBJECTS

We consider a persistent memory containing source memories and LLM-generated derived memories. Each derived node v has a deterministic task rule F_v in the controlled environment, while its stored value y_v is the reader LLM's materialized answer.

3.1 STRUCTURAL DESCENDANTS

Let $G_t^\star$ be the oracle structural graph over memories active immediately before audit time t. An edge $a \to v$ means that a was a rule-required formation-time parent of v. For a root u audited at t,

$$\mathcal{A}_{\mathrm{struct}}(u,t) = \{v : u \rightsquigarrow v \text{ in } G_t^\star\}. \tag{1}$$

This set captures historical influence among currently active stored states.

3.2 AUDIT-TIME UPDATE SENSITIVITY

Let X_{t^-} be the current root assignment immediately before auditing u, including all earlier corrections, and X_{t^+} the assignment just after applying the audit. We define

$$\mathcal{A}_{\mathrm{sens}}(u,t) = \{v \in \mathcal{A}_{\mathrm{struct}}(u,t) : F_v(X_{t^-}) \neq F_v(X_{t^+})\}. \tag{2}$$

This set records which descendants change task-semantic value under the current update. It is intentionally evaluated at audit time rather than formation time; sequential corrections make those notions differ (Appendix C).

3.3 POST-AUDIT STALE STORED STATE

The task value and the value actually materialized by the reader need not agree. We therefore define

$$\mathcal{A}_{\mathrm{stale}}(u,t) = \{v \in \mathcal{A}_{\mathrm{struct}}(u,t) : y_v(t^-) \neq F_v(X_{t^+})\}, \tag{3}$$

where $y_v(t^-)$ is the active stored value immediately before the audit. States already replaced by earlier repairs are excluded. $\mathcal{A}_{\mathrm{sens}}$ and $\mathcal{A}_{\mathrm{stale}}$ are not nested in general: a reader may already be wrong even when the task value does not change, or may already store the value that becomes correct after a sensitive update. We track pre-existing reader errors outside u's structural lineage separately.

Given an observed provenance graph $\widehat{G}$, let $\widehat{A}(u,t)$ be the set reached by the fixed repair operator. For $k \in \{\mathrm{struct}, \mathrm{sens}, \mathrm{stale}\}$ we report

$$C_k(u,t) = \frac{|\widehat{A}(u,t) \cap \mathcal{A}_k(u,t)|}{|\mathcal{A}_k(u,t)|}. \tag{4}$$

Stale-state coverage is an event-level measure. We separately report a full-lifecycle residual-stale fraction: among materialized states that were formed while at least one incorrect ancestor was still unaudited and whose stored value disagrees with the fully corrected task value, it is the fraction never reached by any subsequent relevant repair. We also report absolute post-repair probe error.

3.4 REPAIR ADEQUACY UNDER A FIXED REACHABILITY OPERATOR

The distinction above also clarifies why edge-level provenance scores do not by themselves determine repair quality. For a corrected root u, our repair operator first computes the nodes reachable from u in $\widehat{G}$, invalidates those active nodes, and then re-derives them in topological order. Consequently, two observed graphs that induce the same reached set $\widehat{A}(u,t)$ at every audit make the same invalidation decisions even if their global edge precision, recall, or F1 differ. Conversely, two graphs with similar aggregate edge statistics can differ sharply in repair value if their retained edges connect u to different stale descendants.

We keep the repair operator fixed and vary the provenance representation. Matched random preserves the LLM graph's number of true and false incoming edges node by node and randomizes which true structural parents are retained. This holds structural edge fidelity and budget constant while changing the paths through which a corrected source reaches stale descendants.

Value-agreement support. A rule F_v is *unate* if each parent enters it with a fixed sign, so that F_v is monotone in each parent's signed value. All rules in our testbed are unate: trigger literals, the four L1 families, and $x \wedge y$, $x \vee y$, $x \wedge \neg y$, $\neg x \wedge y$. Given a state X, a parent of v is *value-agreement* if its signed value equals $F_v(X)$; G_X^{va} keeps exactly these edges.

Proposition 1. *Let all rules be unate, let u be the only root whose value changes at audit t, and let every active stored value equal its task value under X_{t^-}. Then every $v \in \mathcal{A}_{\mathrm{sens}}(u,t)$ is reachable from u in $G^{\mathrm{va}}_{X_{t^-}}$.*

Under these conditions, value-agreement support reaches every update-sensitive and stale descendant while retaining only parents aligned with the current conclusion. Appendix C proves the result and relates this audit-time graph to the formation-time graphs used in the experiments.

4 Experimental Setup

4.1 Controlled persistent-memory testbed

We use 50 frozen paired episodes. Each episode contains 24 source claims from six fictional sources, exactly six of which are incorrect. Claims describe eight binary categorical attributes of fictional companies, suppliers, ports, factories, warehouses, carriers, and regulators. One claim arrives per step; every claim is audited 20 steps after arrival, and an incorrect claim is replaced by its known ground-truth value.

The reader operates over a persistent memory. Six L1 queries each depend on four roots through one of four Boolean rule families: $(A \vee B) \wedge (C \vee D)$, $(A \wedge B) \vee (C \wedge D)$, at-least-two-of-four, and at-least-three-of-four. Three base L2 queries combine pairs of L1 conclusions with binary logical rules. The reader receives the natural-language rule and retrieved memory, answers YES/NO, and the answer is written back as a canonical derived memory. L2 retrieval uses a budget of $K = 2$ that prioritizes the two L1 parents; four fixed distractors are included in every query context. After all audits, the same nine questions are issued as write-back-free probes. The oracle graph $G^\star$ records the rule-required parent memories shown to the reader at formation.

4.2 Depth intervention and provenance conditions

For depth $D \in \{1, \ldots, 5\}$, we construct nested variants from the same frozen episode. Roots, incorrect claims, arrival and audit schedules, and the six L1 queries remain fixed. The depth-D chain is a prefix of the depth-5 chain; all chain nodes form at the same step, and unrelated padding memories keep the total number of derived entries fixed across depths. The sweep therefore intervenes on derivation length without shifting formation time or memory count.

We compare five provenance representations under one repair operator. **Oracle structural** uses $G^\star$. **LLM material-support** uses a frozen `deepseek-flash` attributor given the stored conclusion, its question and rule, and candidate memories active at formation; returned valid candidate identifiers become incoming edges. We also compare `gpt-5.6-luna` attribution on the same depth-sweep formations. **Rule-based but-for** retains a true parent only when flipping that parent alone changes the instantiated rule output while all other formation-time inputs are held fixed; overdetermined outputs can therefore retain no individual parent. **Matched random** samples the same number of true and non-true incoming edges as the LLM representation, choosing the true edges uniformly among structural parents. We sample 200 complete random graphs and compute exact repair reachability in each. **Value-agreement support** keeps a true structural parent only when its signed value at formation agrees with the stored conclusion, e.g., the false literals of a stored NO; it uses the task rule and the reader's stored value, but no future audit.

When an audit exposes an incorrect root u, the root is corrected, every active derived memory reachable from u in $\widehat{G}$ is invalidated, and invalidated nodes are re-derived in topological order. Only the provenance graph varies. We report graph precision/recall and edges per node, the three coverage measures, residual stale state, re-derivation calls, and absolute post-repair probe error. Controlled intervals use paired episode-cluster bootstrap resampling.

Table 1: Primary controlled setting.

Setting	Value
Episodes	50 frozen episodes, paired across conditions
Roots	24 per episode (6×4); exactly 6 incorrect; 8 binary attribute types
Derived state	6 L1 nodes + 3 base L2 nodes; depth sweep $D = 1$–5
Lifecycle	One root/step; deterministic audit at arrival $+20$; derived answers persist
Retrieval	Rule-relevant; L2 budget $K = 2$; 4 distractors per query
Readers	`gpt-5.6-luna`, `deepseek-flash`
Attributors	`deepseek-flash` (primary); `gpt-5.6-luna` (same-input depth comparison)
Repair	Reachability invalidation + topological re-derivation
Statistics	Paired episode analysis; episode-cluster bootstrap 95% CI

Information boundary and pairing. Attribution uses formation-time information: the materialized conclusion, its question and rule, and candidate memories active when it was written. Future corrections and their sensitivity, staleness, and probe labels are unavailable to the attributor. Reader outputs, attribution outputs, and base episodes are frozen before lifecycle analysis. Pairing preserves roots, wrong claims, schedules, L1 queries, and chain operations across depths; fixed formation time and memory count isolate derivation length.

4.3 Public behavioral validation

We additionally use MemoryAgentBench Conflict Resolution / FactConsolidation data (Hu et al., 2026). A later fact with the same subject and relation is treated as a correction of the earlier value. Once a pre-correction multi-hop path is complete, we ask the question in-stream and persist the intermediate hop entity; after later corrections, we probe the final question again.

Public repair uses oracle path support: each stored hop-k intermediate records the first k fact keys on the path resolved at formation. OFF disables write-back; S0 retains intermediates; S4 invalidates those supported by a corrected key; S4R additionally re-queries and writes back new intermediates (Appendix D).

The main analysis uses a reversed-order control: for each corrected key, a counterfactual value appears first and the real-world value later acts as the correction, with final gold answers recomputed from the resulting stream. This reduces the advantage of simply reproducing pretrained real-world knowledge. We then separate units by whether the stored intermediate remains valid or becomes invalid after correction. This protocol measures the effect of stored-intermediate validity on post-update answers. Full original/reversed and short/long-context cells are reported in Appendix Fig. 8.

5 Results

5.1 Stale descendants survive source correction

Correcting a root can leave a previously materialized descendant active. Removing the stale derived item improves post-audit accuracy from 0.690 to 0.856, a paired gain of 0.166 (95% CI $[0.111, 0.221]$). Replacing it with unrelated derived text of the same form yields 0.847, while prompt-order changes are smaller (Appendix Fig. 5). The length-matched replacement ties the penalty to stale content. Across audit delay, root-only correction increasingly diverges from closure repair (Appendix Fig. 6): the unrepaired descendants remain available to steer later answers.

5.2 Structural coverage falls with depth while stale-state coverage remains near complete

Figure 1 is the central separation. For Luna, LLM provenance covers 0.68 of structural descendants at $D = 1$ and 0.50 at $D = 5$; for DeepSeek the corresponding values are 0.68 and 0.51. Audit-time sensitive coverage remains 0.95–0.99 for Luna and 1.00 for DeepSeek. More importantly, coverage of actually stale stored descendants remains 0.95–1.00 for Luna and 1.00 throughout for DeepSeek.

The repair outcome tracks the latter separation. Across the complete audit sequence, the residual-stale fraction is 0–2.3% for Luna and 0 for DeepSeek. At $D = 5$, post-repair probe error is 0.10 errors/episode for LLM provenance versus 0.04 for oracle and 0.94 with no repair on Luna; DeepSeek is $0.00/0.00/0.86$. At depth five, repair remains near oracle with structural descendant coverage of about one half. Full reader-by-depth confidence intervals are in Appendix Fig. 4 and Table 3.

5.3 Reader errors separate update sensitivity from stored-state staleness

On active stored states in the LLM-graph replay, $\mathcal{A}_{\text{sens}}$ and $\mathcal{A}_{\text{stale}}$ coincide exactly for DeepSeek (682 root–descendant pairs pooled over the depth sweep; 274 in the base setting). For Luna, 18 pairs are sensitive but fresh and 17 are stale but insensitive, giving 35 disagreements over depth; the oracle-graph replay gives 28. Both graphs give 7 disagreements in the base setting. Every disagreement traces to a reader error, including an answer that inherited a wrong ancestor; none comes from stale state left by a missed earlier repair. Appendix A traces a stale-but-insensitive case. Structural descendants outnumber sensitive and stale states roughly fourfold (2,651 versus 682 and 681 pairs for Luna under the LLM graph). Table 6 reports both set differences and out-of-lineage errors; Appendix C specifies the shared formation set used by the graph replays.

5.4 Equal structural edge fidelity can yield very different repair value

Figure 2 tests whether the result is merely a consequence of using a sparse graph. In the pooled depth sweep, LLM provenance has about 2.95–2.98 edges per node with structural precision 0.79 and recall 0.69. The matched-random graph is constructed with the same edge count and the same numbers of true and non-true edges, so its aggregate structural precision/recall is the same. Yet stale-state coverage is $0.98/1.00$ for LLM provenance and only $0.66/0.68$ for matched random (Luna/DeepSeek). Corresponding residual-stale fractions are $0.012/0.000$ versus $0.23/0.21$.

A symbolic but-for graph provides a different selectivity test. It has perfect edge precision but only 0.42–0.43 recall, covering $0.83/0.85$ of stale state and leaving $0.17/0.14$ residual stale. The retained parent identities determine which stale states the fixed repair operator reaches. LLM provenance also reduces repair work relative to oracle structural closure, averaging about 5.9–6.0 re-derivations per episode versus 8.2 for oracle in the pooled depth analysis.

Overdetermined support becomes relevant after later corrections. The but-for control keeps a true parent only when it alone decides the formation-time output. In an overdetermined Boolean state, which covers about 21% of formations, no single parent is decisive although several are genuine supports that can become decisive after later corrections, so dropping them removes paths needed by future updates. For unate rules with correct stored answers, every decisive parent is value-agreement; the but-for graph drops the individually redundant supports retained by value-agreement.

Value-agreement support reproduces the LLM result. The value-agreement graph has perfect edge precision and recall 0.68, close to the LLM graph's 0.69, and shares 69–72% of its edges (Jaccard). It reaches $0.971/1.00$ of stale state, leaves $0.018/0$ residual stale, and performs $5.7/5.8$ re-derivations per episode, against $0.977/1.00$, $0.012/0$, and $5.9/6.0$ for LLM provenance (Luna/DeepSeek, pooled depth sweep). All 13 residual stale states it leaves (Luna) trace to reader errors: 4 stored wrong at formation and 9 inherited from a wrong ancestor. Value-agreement support explains the selective repair achieved by material-support attribution in this testbed.

Higher structural fidelity increases repair work. Replacing the `deepseek-flash` attributor with `gpt-5.6-luna` on the same formations yields a structurally better graph: edge precision/recall rise from $0.79/0.69$ to $0.90/0.79$, and structural coverage rises by 0.14–0.15. Stale-state coverage does not rise ($0.973/1.00$ versus $0.977/1.00$; difference -0.004 $[-0.011, 0.000]$), while re-derivations rise by 1.0 $[0.7, 1.3]$ per episode (Appendix Table 5). Both attributors retain 95–96% of value-agreement edges; they differ in true edges that point against the stored conclusion, of which Luna retains 42–43% and DeepSeek 11–12%. The additional true edges increase recomputation without improving stale-state coverage.

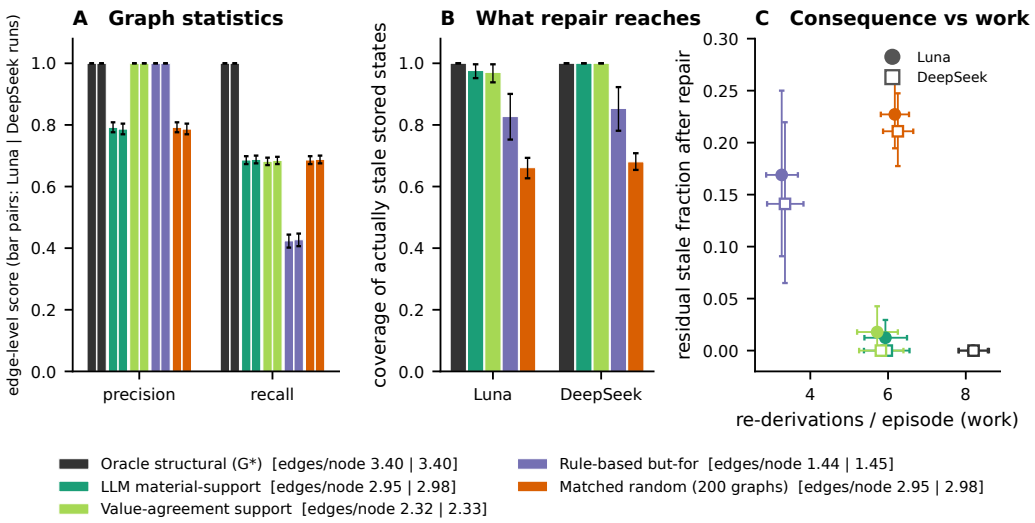

Figure 2: **Comparable structural edge statistics can have very different repair value. A:** Structural edge statistics and edge budget for oracle, LLM material-support, value-agreement, symbolic but-for, and edge-matched random provenance. **B:** Stale-state coverage under the fixed reachability repair: LLM $0.98/1.00$, value-agreement $0.97/1.00$, but-for $0.83/0.85$, matched random $0.66/0.68$ (Luna/DeepSeek). **C:** Residual stale state versus re-derivation work. Matched-random results use 200 sampled graphs with exact reachability; reader answers are not re-generated for graph-only controls. Error bars are episode-cluster paired bootstrap 95% CIs.

Matched random uses 6.2–6.3 re-derivations per episode, close to the LLM graph's 5.9–6.0, and leaves about one fifth of lifecycle-stale state unreached. At comparable repair work, the LLM graph reaches substantially more stale descendants.

5.5 Public multi-hop memory exhibits the same validity boundary behaviorally

On the primary reversed-order multi-hop split, retaining a stored intermediate that later becomes invalid lowers post-update accuracy from 0.921 to 0.553 for Luna and from 0.974 to 0.763 for DeepSeek (Fig. 3). Closure repair returns both readers to the corresponding no-write-back level. When the intermediate remains valid, the stale-write-back arm differs from no write-back by $+0.00$ for Luna and $+0.04$ for DeepSeek. The resulting write-back $\times$ validity interaction is -0.37 (95% CI $[-0.53, -0.22]$) for Luna and -0.25 ($[-0.41, -0.11]$) for DeepSeek. The behavioral penalty is therefore concentrated in units where the persistent intermediate actually becomes invalid after correction.

In the longer-context reversed control, the interaction is -0.34 $[-0.52, -0.18]$ for Luna and -0.07 $[-0.18, 0.00]$ for DeepSeek. Original-order cells have larger negative interactions and restore the pretrained-world-knowledge asymmetry addressed by reversal. We use the reversed short-context cell as the primary public result and report each ordering and context condition separately (Appendix Fig. 8 and Table 7).

6 Discussion

The stored conclusion determines useful support. The two attributors retain 95–96% of value-agreement edges. Their main difference lies in parents whose values point against the stored conclusion: retaining more of these expands the repair closure and increases recomputation, with nearly identical stale-state coverage. The symbolic control reproduces this selectivity from the rule and stored answer, identifying agreement with the materialized conclusion as the mechanism behind the LLM graph's repair value.

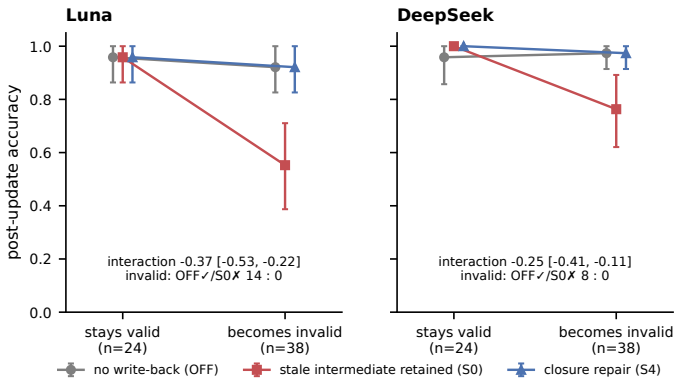

Figure 3: **Persistent intermediates harm behavior when they become invalid.** MemoryAgentBench FactConsolidation multi-hop, reversed-order short-context control. We compare no write-back (OFF), retaining the stored intermediate (S0), and oracle-path closure invalidation (S4), separately for units whose intermediate remains valid or becomes invalid after correction. $n = 24$ valid and 38 invalid units per reader. Error bars are unit-cluster paired bootstrap 95% CIs.

Repair relevance changes across the lifecycle. A parent can be redundant at formation and become decisive after other sources are corrected. The but-for graph drops such parents; value-agreement retains paths for later repair. Reader errors create another repair target: an answer may remain wrong even when its task value is unchanged by the correction. Repair relevance therefore evolves with both the stored value and the history of corrections and repairs.

Evaluate reached states alongside recovered edges. Alongside structural precision and recall, repair evaluation should measure stale descendants reached, residual stale state, re-derivation work, and post-repair answer error. Matched random has equal structural fidelity and comparable repair work yet reaches far less stale state. The second attributor recovers more edges and adds work while stale coverage stays flat. These comparisons establish separate structural and lifecycle axes for evaluating provenance.

7 Limitations

The controlled testbed measures value-level validity through deterministic binary rules, and the public benchmark measures the validity of stored multi-hop intermediate entities. Free-text memories can also contain outdated rationales or factual details whose revision leaves the final task answer unchanged. The reachability guarantee assumes unate rules and correct stored values; parity and free-text generation require other definitions of support. Audits correct one root at a time; simultaneous corrections require evaluating sensitivity and staleness against the joint updated state. The evaluated root-triggered graphs leave pre-existing errors outside the corrected root's lineage unrepaired (67 Luna root–state pairs in the depth replay; Appendix C).

8 Conclusion

Persistent memory can remain wrong after its source is corrected, and the state that needs repair is much smaller than its structural lineage. LLM material-support provenance covers half to two-thirds of structural descendants while reaching 0.95–1.00 of stale state. Equal structural fidelity produces sharply different stale coverage, and greater fidelity adds recomputation without improving coverage. Value-agreement support reproduces the LLM result and admits a reachability guarantee for unate rules with correct stored values. Provenance quality therefore has two axes: the lineage recovered and the stale materialized state repaired.

Reproducibility statement

The controlled testbed uses frozen paired episodes and deterministic lifecycle schedules. Appendix A specifies the source claims, rule families, depth construction, provenance conditions, repair operator, model configurations, and statistical procedure. The anonymous supplementary archive contains the frozen episode definitions, lifecycle labels, final figure data and plotting scripts, public-benchmark intervention data, prompt templates for the reported attribution and public-memory evaluations, model configurations, and analysis snapshots used to audit the reported numbers.

AI use statement

Generative AI tools were used to refine research hypotheses, provide feedback on experimental methodology, assist implementation and analysis, help formulate mathematical claims and develop their proofs, support interpretation of results, search and summarize relevant literature, draft and edit manuscript text, and create or modify scientific figures. AI-assisted code, quantitative analyses, citations, claims, and figures were reviewed and checked by the authors against the underlying experiments and source material. The authors made the final scientific judgments and take responsibility for all content and artifacts in this submission.

References

Umut A. Acar, Guy E. Blelloch, and Robert Harper. Adaptive functional programming. In *POPL*, 2002.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *International Conference on Database Theory (ICDT)*, pages 316–330, 2001.

James Cheney, Amal Ahmed, and Umut A. Acar. Provenance as dependency analysis. In *Database Programming Languages (DBPL)*, pages 138–152, 2007.

Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. In *ECAI*, pages 712–720, 2020.

Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

Ashish Gupta, Inderpal S. Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *SIGMOD*, 1993.

Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *The British Journal for the Philosophy of Science*, 56(4):843–887, 2005.

Matthew A. Hammer, Khoo Yit Phang, Michael Hicks, and Jeffrey S. Foster. Adapton: Composable, demand-driven incremental computation. In *PLDI*, 2014.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *ICLR*, 2026. MemoryAgentBench; arXiv:2507.05257.

Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proceedings of the VLDB Endowment*, 4(1):34–45, 2010.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv:2310.08560, 2023.

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 *UIST*, 2023.

Ruoling Qi, Xuaner Wu, Penghang Liu, Jian Chen, and Yirui Liu. REVISE: Validity-guided recovery for online revisions in agent workflows. arXiv:2609.00643, 2026.

Yi Ting Shen, Kentaroh Toyoda, and Alex Leung. Revoked but still authoritative: An empirical study of revocation enforcement in agent-memory systems. arXiv:2609.08258, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *NeurIPS*, 2025.

Caili Yu, Yiqi Wang, Jiaqi Zhang, Yiqun Duan, Mingkai Zheng, Zhangkai Wu, Kaize Shi, and Taotao Cai. From faulty memories to corrected actions: Dependency-guided rollback repair for memory-augmented agents. arXiv:2608.10502, 2026.

Yang Zhao, Chengxiao Dai, Mengying Kou, and Yue Xiu. MEMOREPAIR: Barrier-first cascade repair in agentic memory. arXiv:2605.07242, 2026.

Yue Zhao. GRADE: Graph representation of LLM agent dependency and execution. arXiv:2606.22741, 2026.

A Controlled Testbed and Reproducibility Details

Episode generation. Episodes are procedurally generated by the authors from integer seeds (generator `gen_v1`, included in the supplementary archive) and written to disk before any model call; they are never regenerated per condition. Generation is purely programmatic: no language model is involved, so there is no generation prompt, and every natural-language string the reader sees is filled into the fixed templates given under *Prompt templates* below. Entity names are invented by concatenating two nonsense syllables with a type suffix (e.g., *Quendris Industries*, *Pelldell Port*), so no entity or attribute value can be recalled from pretraining. We generated 200 seeds and use the first 50 (`ep0000`–`ep0049`) in every reported controlled result; the remaining episodes were not used. Every released file is byte-identical to the generator output for its seed.

Source claims and schedule. Each episode contains 24 roots from six fictional sources, four claims per source. The eight attribute types are binary: company revenue trend (up/down), company margin trend (up/down), supplier delivery status (on_time/delayed), port operating status (open/disrupted), factory production status (normal/reduced), warehouse inventory level (adequate/low), carrier capacity (available/constrained), and regulator permit status (approved/pending). Each root's true value is drawn uniformly, and roots are randomly assigned to source slots. Exactly six roots are incorrect and store the opposite of their ground-truth value. The LOW placement takes one incorrect root from each of the six sources; the HIGH placement takes three incorrect roots from each of two sources. Claims arrive one per step from $t = 0$ to 23, round-robin over a shuffled source order. Every root is audited 20 steps after arrival; an incorrect root is then replaced by its true value. Post-audit probes are issued at $t = 46$, after the last audit.

Derived-state rules. The six L1 nodes partition the 24 roots at random, four parents per node. Each parent enters as a literal that tests whether the root takes a sampled trigger value (e.g., "the port is *open*"). Rule families are $(A \vee B) \wedge (C \vee D)$, $(A \wedge B) \vee (C \wedge D)$, at-least-two-of-four, and at-least-three-of-four, sampled uniformly (82/75/71/72 of the 300 L1 nodes in the 50 used episodes). The three base L2 nodes each combine two L1 conclusions with one of $x \wedge y$, $x \vee y$, $x \wedge \neg y$, $\neg x \wedge y$. Generation rejection-samples until exactly three of six L1 nodes are true and the three L2 nodes are not all equal under the true roots. An L1 query is asked one step after its last parent arrives plus a random lag of 0–2 steps; an L2 query is asked after both L1 parents in the same way. Each episode also holds 12 background facts about unrelated entities, drawn from four attribute types disjoint from the root attributes (headcount trend, audit rating, credit outlook, fleet status); each

query context includes four of them as distractors that never support any node. The reader receives a natural-language rendering of the task rule and retrieved memories, returns YES/NO, and every valid in-stream answer is written back as a canonical derived memory.

Oracle structural graph and lifecycle labels. For a materialized node v, $G^\star$ connects v to rule-required memories shown at formation. Structural descendants are computed transitively over active derived memories. At each incorrect-root audit, sensitivity is recomputed using the current root assignment immediately before and after that correction. Staleness uses the active reader value immediately before audit and the post-audit task value. States already replaced by earlier repairs are excluded.

Depth construction. The depth sweep constructs a nested chain from the same six L1 nodes. The first leaf $L1_a$ is the L1 node containing the incorrect root whose audit arrives last, so that an uncorrected error is present when the chain forms; the next four L1 nodes by index are the remaining leaves. The chain is $C_2 = \mathrm{op}_2(L1_a, L1_b)$ and $C_k = \mathrm{op}_k(C_{k-1}, L1_k)$ for $k = 3, \ldots, 5$, with operations drawn from the four L2 rules. Chain operations are sampled once per base episode and shared across depth variants, so the depth-D graph is a prefix of the depth-5 graph; $D = 1$ queries $L1_a$ alone. All chain nodes form at the same step, one step after the last L1 query, and $5 - D$ unrelated padding entries keep the number of derived memories at ten across depth. The original L2 nodes are removed in depth variants. Roots, wrong claims, arrival and audit schedules, and L1 queries are paired across all depths.

Worked example. Table 2 shows node L1_5 of `ep0017` under LOW placement in the base setting. Its question asks whether Project Volans is *launch ready*, with rule at-least-two-of-four over the literals. The node is queried at $t = 11$, one step after its last parent r01 arrives with the incorrect claim *down*. Literals C and D hold, so the task value at formation is YES.

Table 2: Parents of node L1_5 (`ep0017`, LOW). Only r01 is an incorrect claim.

Literal	Root	Entity	Attribute	Trigger	True	Claimed	Arrival / correction
A	r01	Rheltane Industries	margin trend	up	up	**down**	10 / 30
B	r03	Ulmdell Port	operating status	disrupted	open	open	1 / –
C	r14	Rheloth Freight	capacity	constrained	constrained	constrained	9 / –
D	r17	Galdoth Industries	margin trend	down	down	down	3 / –

The Luna reader nevertheless stored NO. The material-support attributor, given this stored conclusion, returned r01 and r03, the two false literals that support NO; the value-agreement graph contains exactly the same edges. The audit of r01 at $t = 30$ makes literal A true, but three literals then hold instead of two, so the task value stays YES and L1_5 $\notin \mathcal{A}_{\mathrm{sens}}(\mathrm{r01}, 30)$. The stored NO differs from the post-audit value YES, so L1_5 $\in \mathcal{A}_{\mathrm{stale}}(\mathrm{r01}, 30)$: a stale-but-insensitive pair caused by a reader error. Because the attributor had linked r01 to L1_5, the repair reached the node, the reader re-derived YES with new support $\{\mathrm{r01}, \mathrm{r14}, \mathrm{r17}\}$, and the post-audit probe at $t = 46$ was answered correctly. The case also shows why conditioning on the stored value matters: a graph built from the task value YES would link r14 and r17 instead, and would not reach this error. The released episode files contain the task structure only; reader answers and attributor outputs come from the frozen runs and are provided separately in the supplementary archive.

Material-support attribution. For each stored derived conclusion, the attributor receives the conclusion, its question and decision rule, and candidate memories active at formation. The fixed instruction asks for candidate identifiers that materially support the conclusion. Only valid candidate identifiers are retained as incoming edges. The attributor does not receive future corrections, audit times, corrected root values, sensitivity labels, stale-state labels, or post-audit outcomes.

Rule and random controls. For the but-for graph, each true formation-time parent is flipped alone while all other formation-time inputs are fixed; its edge is retained exactly when the deterministic rule output changes. About 21% of formations are overdetermined, in which case no individual parent is locally decisive. For matched random, the number of true and non-true incoming edges is copied from the LLM graph for each node, true edges are sampled uniformly without replacement, and exact reachability is evaluated over 200 sampled complete graphs.

Model configurations. The controlled readers are `gpt-5.6-luna` and `deepseek-flash`, both through direct APIs for the main experiments. Luna uses low reasoning effort and strict JSON output; DeepSeek uses temperature 0 and JSON output. Both allow 4,000 output tokens and one 32,000-token retry. The stale-derived ablation uses Luna through the Codex CLI. The primary material-support attributor is `deepseek-flash` with thinking enabled, JSON output, a 4,000-token limit, and one 32,000-token rescue. The second attributor is `gpt-5.6-luna` through the Codex CLI with low reasoning effort and a structured support schema. It receives the same recorded conclusion, rule, and candidate memories, with no explicit output-token cap. Reader and attributor outputs are frozen before analysis.

Statistics. Controlled confidence intervals use paired episode-cluster bootstrap resampling with 4,000 draws; all series compared within a reader/depth cell share the same resampled episode indices. Public intervals use unit-cluster paired bootstrap resampling. Random-graph summaries replay exact reachability on logged formations rather than regenerating reader answers.

Prompt templates. Templates are reproduced as the code builds them; rendered prompts and raw outputs for one full trajectory and for the worked example are in the supplementary archive. The controlled reader receives the following system prompt.

```
You are the reasoning module of a supply-chain monitoring assistant
that keeps a persistent memory. Answer the question using the decision
rule and the memory entries provided. Memory entries are the
assistant's current stored beliefs; they may be incomplete. If a needed
fact is missing, make your best judgment from what is available.
Return JSON with fields: answer ("YES" or "NO"), confidence (0.0-1.0),
used_memory_ids (list of memory ids you relied on).
```

The user message gives the current time, the question "is {name}({subject}) YES or NO?", the decision rule (for L2 and chain nodes, followed by each parent's rule), and the memory entries. Root entries read "{entity} {attribute} is {value}." with the currently held value, derived entries read "Derived fact: {name}({subject}) = YES—NO.", and distractors use the root format. Entries are ordered by a fixed hash of episode, node, and memory id, so the order is identical across conditions. An L1 rule reads, e.g., "{name}({subject}) = YES if and only if (L1 OR L2) AND (L3 OR L4). Otherwise NO.", where each literal is "[{entity} {attribute} is {trigger value}]". An answer that is not YES or NO is scored incorrect and never written back. The material-support attributor receives the following system prompt.

```
You audit the provenance of a derived conclusion stored in an
assistant's memory. You are given the conclusion, the question and
decision rule it answered, and the candidate memory items that existed
when it was recorded. Identify which candidate items materially support
the conclusion (items whose content the conclusion depends on). Return
JSON: {"supports": [{"id": "<candidate id>", "confidence": <0-1>}]}.
attr_v1
```

For Luna attribution, the same system content is enclosed in an `<instructions>` block, followed by the user content and the instruction "Answer directly; do not run any commands."

The attributor user message gives the recorded conclusion with its time, the question, the decision rule, and every active memory item at recording time except the conclusion itself, sorted by id. Returned ids are intersected with the candidates, confidences are ignored, and an unresolved attribution is never treated as empty support. The public-benchmark reader prompt asks for a short entity answer and intermediate entities per hop; the reported public repair conditions use the oracle path support defined in Appendix D.

B Full Controlled Results

Figure 4 and Table 3 report the complete reader-by-depth sweep. Subsequent figures and tables give the content ablation, delay sweep, provenance controls, and second-attributor comparison.

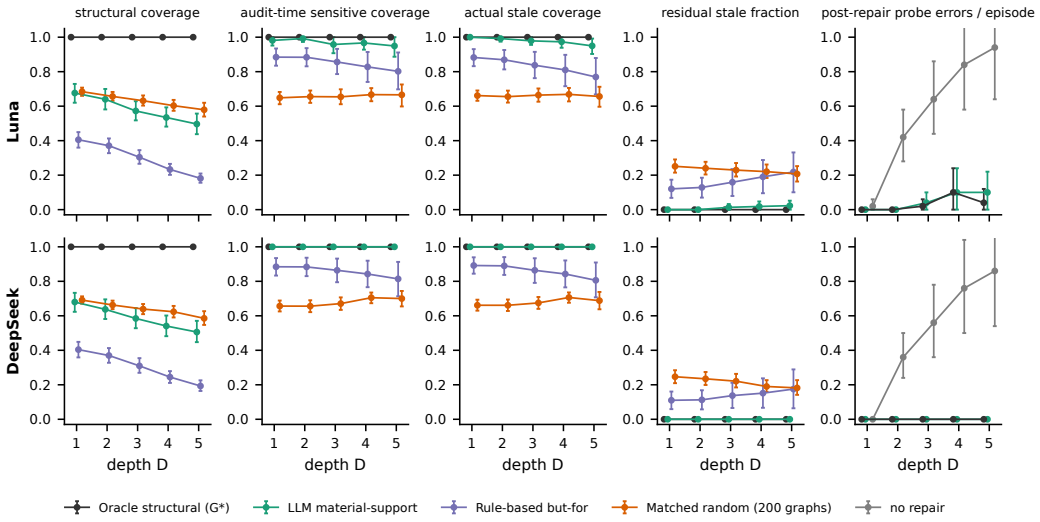

Figure 4: **Full reader-by-depth results.** Structural, audit-time sensitive, and stale-state coverage, residual stale state, and post-repair probe error for the depth intervention. The full provenance set is shown for repair-level metrics. Error bars are episode-cluster paired bootstrap 95% CIs over 50 episodes per reader/depth.

Table 3: Reader-specific LLM-provenance depth sweep. Brackets are episode-cluster bootstrap 95% CIs.

Reader	D	Struct. cov. (LLM)	Sens. cov. (LLM)	Stale cov. (LLM)	Residual stale (LLM)	Probe errors / episode		
						no repair	LLM	oracle
Luna	1	0.68 [0.62, 0.73]	0.98 [0.95, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.02	0.00	0.00
	2	0.64 [0.58, 0.70]	0.99 [0.97, 1.00]	0.99 [0.97, 1.00]	0.00 [0.00, 0.00]	0.42	0.00	0.00
	3	0.57 [0.52, 0.63]	0.96 [0.91, 1.00]	0.98 [0.96, 1.00]	0.01 [0.00, 0.03]	0.64	0.04	0.02
	4	0.53 [0.48, 0.59]	0.97 [0.93, 1.00]	0.97 [0.94, 1.00]	0.02 [0.00, 0.05]	0.84	0.10	0.10
	5	0.50 [0.44, 0.56]	0.95 [0.89, 1.00]	0.95 [0.90, 0.99]	0.02 [0.00, 0.05]	0.94	0.10	0.04
DeepSeek	1	0.68 [0.62, 0.73]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.00	0.00	0.00
	2	0.64 [0.58, 0.70]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.36	0.00	0.00
	3	0.58 [0.53, 0.64]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.56	0.00	0.00
	4	0.54 [0.48, 0.60]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.76	0.00	0.00
	5	0.51 [0.45, 0.57]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	0.86	0.00	0.00

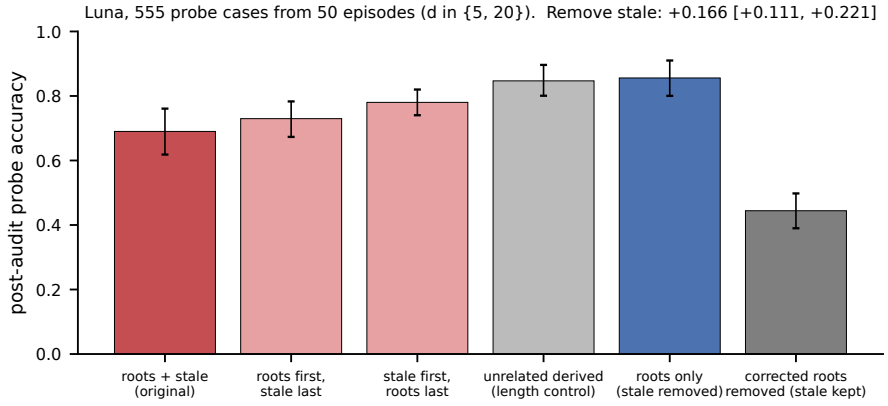

Figure 5: **Stale-derived content is causally harmful in the controlled probes.** Removing the stale derived item improves accuracy from 0.690 to 0.856 (paired difference 0.166 [0.111, 0.221]); replacing it with an unrelated derived item of the same form yields 0.847. Prompt-order changes are smaller. There are 555 post-audit probe cases from 50 episodes; intervals are paired episode-cluster bootstrap 95% CIs.

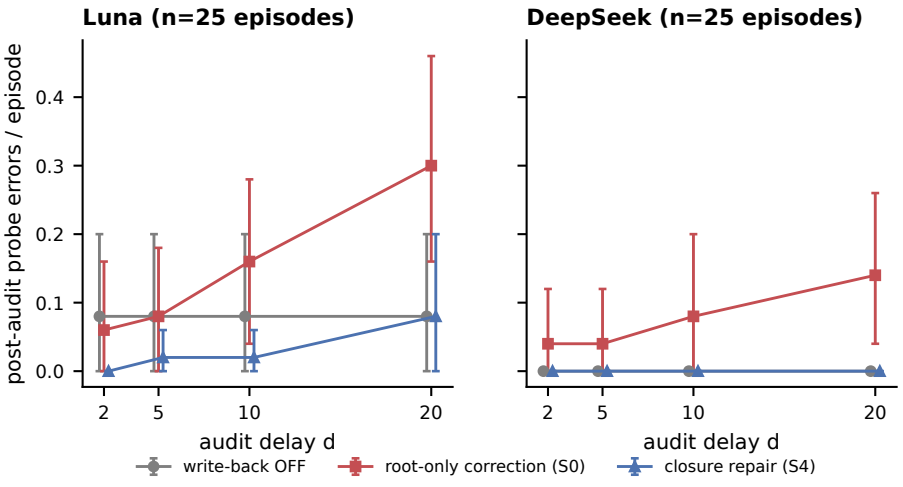

Figure 6: **Source-only correction diverges from closure repair as audit delay grows.** Post-audit probe errors per episode for no write-back, root-only correction, and provenance-closure repair on the 25 episodes evaluated by both readers. At delay 20, root-only minus closure error is 0.22 [0.04, 0.40] for Luna and 0.14 [0.04, 0.26] for DeepSeek.

Table 4: Provenance graph statistics and repair outcomes. The first block pools the depth sweep within episode; the second reports the base $K = 2$ setting.

Setting	Reader	Provenance	Edges/node	Edge P	Edge R	Stale cov.	Residual stale	Re-deriv./ep.
depth sweep D1-5	Luna	Oracle structural ($G^\star$)	3.40	1.00	1.00	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	8.2
		LLM material-support	2.95	0.79	0.69	0.98 [0.95, 1.00]	0.01 [0.00, 0.03]	5.9
		Value-agreement support	2.32	1.00	0.68	0.97 [0.94, 1.00]	0.02 [0.00, 0.04]	5.7
		Rule-based but-for	1.44	1.00	0.42	0.83 [0.75, 0.90]	0.17 [0.09, 0.25]	3.3
		Matched random	2.95	0.79	0.69	0.66 [0.63, 0.69]	0.23 [0.19, 0.26]	6.2
	DeepSeek	Oracle structural ($G^\star$)	3.40	1.00	1.00	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	8.2
		LLM material-support	2.98	0.79	0.69	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	6.0
		Value-agreement support	2.33	1.00	0.68	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	5.8
		Rule-based but-for	1.45	1.00	0.43	0.85 [0.78, 0.92]	0.14 [0.06, 0.22]	3.3
		Matched random	2.98	0.79	0.69	0.68 [0.65, 0.71]	0.21 [0.18, 0.25]	6.3
main setting	Luna	Oracle structural ($G^\star$)	3.29	1.00	1.00	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	8.6
		LLM material-support	3.04	0.76	0.70	1.00 [0.99, 1.00]	0.00 [0.00, 0.01]	6.1
		Value-agreement support	2.25	1.00	0.68	1.00 [0.99, 1.00]	0.00 [0.00, 0.01]	5.8
		Rule-based but-for	1.41	1.00	0.43	0.85 [0.80, 0.90]	0.15 [0.10, 0.19]	3.2
		Matched random	3.04	0.76	0.70	0.70 [0.67, 0.72]	0.22 [0.19, 0.24]	6.5
	DeepSeek	Oracle structural ($G^\star$)	3.29	1.00	1.00	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	8.6
		LLM material-support	3.07	0.75	0.70	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	6.1
		Value-agreement support	2.26	1.00	0.69	1.00 [1.00, 1.00]	0.00 [0.00, 0.00]	5.8
		Rule-based but-for	1.42	1.00	0.43	0.88 [0.84, 0.92]	0.12 [0.08, 0.17]	3.2
		Matched random	3.07	0.75	0.70	0.70 [0.67, 0.73]	0.21 [0.19, 0.24]	6.5

Table 5: Second attributor on identical formations (depth sweep, $D = 1$–5 pooled). Rows are grouped by the reader whose runs supplied the formations; the difference is `gpt-5.6-luna` minus `deepseek-flash`, with episode-cluster paired bootstrap 95% CIs over 50 episodes. The two attributors share 71% of edges (Jaccard), and each shares 72–73% with value-agreement support.

Reader runs	Metric	DeepSeek attributor	Luna attributor	Difference [95% CI]
Luna	Edge precision	0.79	0.90	$+0.11$ [0.09, 0.13]
	Edge recall	0.69	0.79	$+0.10$ [0.09, 0.12]
	Structural coverage	0.56	0.70	$+0.14$ [0.11, 0.18]
	Sensitive coverage	0.968	0.965	-0.003 [-0.014, 0.007]
	Stale coverage	0.977	0.973	-0.004 [-0.011, 0.000]
	Residual stale	0.012	0.021	$+0.008$ [0.000, 0.021]
	Re-derivations / episode	5.93	6.92	$+0.98$ [0.69, 1.30]
DeepSeek	Edge precision	0.79	0.90	$+0.12$ [0.10, 0.14]
	Edge recall	0.69	0.79	$+0.10$ [0.09, 0.12]
	Structural coverage	0.57	0.72	$+0.15$ [0.11, 0.19]
	Sensitive / stale coverage	1.00	1.00	0.000 [0.000, 0.000]
	Residual stale	0.000	0.000	0.000 [0.000, 0.000]
	Re-derivations / episode	5.97	7.00	$+1.03$ [0.74, 1.33]

Recall of value-agreement edges: 0.95 (DeepSeek attributor), 0.96 (Luna attributor). Recall of true edges pointing against the stored conclusion: 0.11–0.12 (DeepSeek attributor), 0.42–0.43 (Luna attributor).

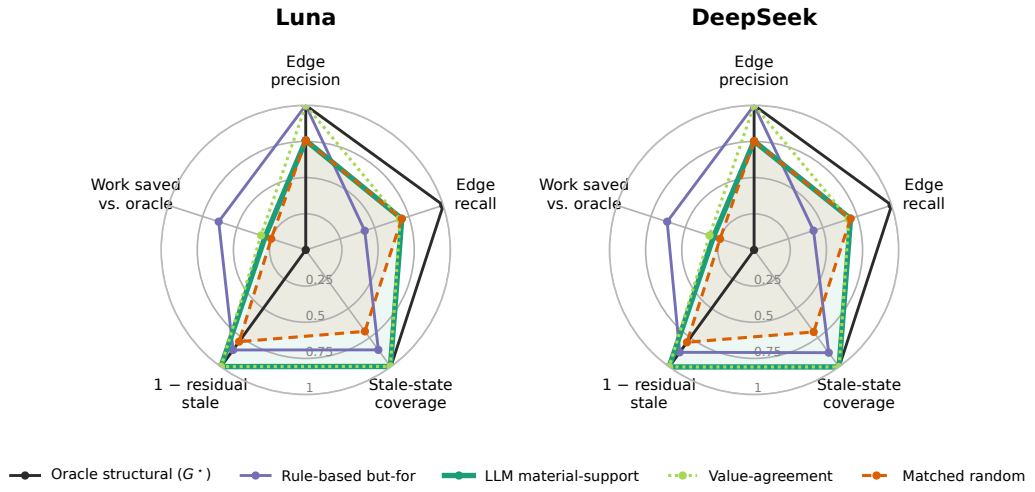

Figure 7: **Provenance profiles in the base $K = 2$ setting** (Table 4, second block). LLM material-support and matched random coincide on edge precision and recall by construction, yet separate on stale-state coverage and residual stale state; value-agreement support nearly traces the LLM profile with perfect precision; but-for gains precision and work savings at the cost of recall and coverage. "Work saved" is $1 - (\text{re-derivations per episode})/8.64$, relative to oracle structural closure, so oracle is 0 on that axis.

C Definition and Robustness Checks

Proof of Proposition 1. Induct over a topological order of u's descendants. Let $v \in \mathcal{A}_{\text{sens}}(u,t)$ and write x, x' for the signed parent values of v under X_{t^-} and X_{t^+}, so that F_v is nondecreasing in each coordinate. Suppose $F_v(x) = 1$; the case $F_v(x) = 0$ is symmetric. If every parent that changes had $x_p = 0$, all changes would go from 0 to 1, so $x' \geq x$ and monotonicity would give $F_v(x') \geq F_v(x) = 1$, contradicting sensitivity. Hence some changed parent p has $x_p = 1 = F_v(x)$, i.e., p is value-agreement. Because u is the only root that changes, p is either u or a derived node whose task value changes, i.e., $p \in \mathcal{A}_{\text{sens}}(u,t)$, which is reachable from u by induction. Since stored values equal task values under X_{t^-}, value-agreement on stored values and on task values coincide. □

Connection to minimal witnesses and sufficient reasons. Write a unate local rule as a monotone Boolean function $g(z)$ of signed parent values, and let its correct output be $b = g(z)$. A sufficient reason is a subset-minimal set W of coordinates such that fixing z_W forces output b under every completion of the other coordinates (Ignatiev et al., 2019; Darwiche and Hirth, 2020). For positive outputs of monotone Boolean provenance, these sets are minimal witnesses (Buneman et al., 2001). Let $V_b(z) = \{i : z_i = b\}$ be the value-agreement set and $\mathcal{W}_b(z)$ the family of sufficient reasons. Then

$$\bigcup_{W \in \mathcal{W}_b(z)} W \;\subseteq\; V_b(z).$$

To see this for $b = 1$, suppose a sufficient reason fixes a coordinate to 0. Removing that coordinate preserves sufficiency: completions setting it to 0 already produce 1, and setting it to 1 cannot lower a monotone output. Such a coordinate cannot occur in a minimal reason. For $b = 0$, the symmetric argument removes coordinates fixed to 1. The inclusion can be strict: for $g(z) = z_1 \vee (z_2 \wedge z_3)$ at $z = (1,1,0)$, the only sufficient reason is $\{1\}$, while value-agreement retains $\{1,2\}$. Value-agreement therefore retains a superset of all minimal sufficient reasons without enumerating them. This connection applies when the stored conclusion equals $g(z)$; the experimental graph also conditions on erroneous stored conclusions, as in the worked example in Appendix A.

Formation-time edges. The experiments build value-agreement edges at formation, whereas Proposition 1 uses values just before the audit. Under oracle closure every audit of a structural ancestor re-derives the state, so no stored state has an ancestor audited after its formation; the two constructions then coincide, and formation-time and audit-time sensitivity labels agree on all stored states of oracle trajectories. On LLM-provenance trajectories they disagree for 2.7–4.5% of stored pairs.

Audit-time versus formation-time sensitivity. Once several corrections occur sequentially, a root's effect at formation need not equal its effect when it is audited. A formation-time single-root label assigns no sensitive root to $104/737$ (14%) of the descendants that are stale under the final corrected lifecycle in the depth sweep, and to $36/288$ (12.5%) in the base setting; these require two or more corrections before their value changes. Main-text sensitivity therefore always uses Eq. (2) at the actual audit state.

Sensitivity and actual staleness. We replay the oracle and LLM graphs on the same frozen formations from LLM-attribution runs, removing each formation after the first repair that reaches it. Thus each pair uses a state still active immediately before the current correction. The two sets coincide for DeepSeek. For Luna, the LLM graph gives 18 sensitive-but-fresh and 17 stale-but-insensitive pairs over depth (35 in total); the oracle graph gives 17 and 11 (28 in total). Both give 3 and 4 in the base setting (7 in total). All disagreements arise from values that were wrong when written, including inherited reader errors. Stale-but-insensitive states left by a missed earlier repair number zero under both graphs (Table 6).

Structural-coverage denominators. The structural-coverage gap persists under all three denominators. Pooled over depth, LLM provenance covers $0.559/0.567$ (Luna/DeepSeek) of structural pairs active under its own graph, $0.580/0.590$ of those active under oracle-graph replay, and $0.563/0.571$ of all pairs formed before the correction. The oracle denominator is held fixed across the evaluated graphs. Base-setting values are 0.61–0.62 under all three.

Table 6: Structural, audit-time sensitive, and post-audit stale root–descendant pairs, counted over active states in oracle and LLM reachability replays of the same logged formations. Stale-but-insensitive pairs are split by cause: the stored value was wrong when written (reader error) or an earlier repair was missed. Out-of-lineage stale states are tracked separately and are not charged to the audited root's repair.

Setting	Reader	Replay graph	$\lvert A_{\mathrm{struct}}\rvert$	$\lvert A_{\mathrm{sens}}\rvert$	$\lvert A_{\mathrm{stale}}\rvert$	both	sens\stale	stale\sens (reader err. / earlier audit)	out-of-lineage stale
depth sweep D1–5	Luna	oracle	2047	614	608	597	17	11 / 0	55
depth sweep D1–5	Luna	llm	2651	682	681	664	18	17 / 0	67
depth sweep D1–5	DeepSeek	oracle	2050	621	621	621	0	0 / 0	0
depth sweep D1–5	DeepSeek	llm	2630	682	682	682	0	0 / 0	0
main setting	Luna	oracle	864	251	252	248	3	4 / 0	34
main setting	Luna	llm	997	274	275	271	3	4 / 0	34
main setting	DeepSeek	oracle	864	251	251	251	0	0 / 0	0
main setting	DeepSeek	llm	997	274	274	274	0	0 / 0	0

Out-of-lineage reader errors. Event-specific repair is rooted at the corrected source u. We report pre-existing errors outside u's structural lineage separately from repair targets within that lineage. In the LLM-graph replay, Luna has 67 such root–state pairs pooled over depth and 34 in the base setting; DeepSeek has none. The evaluated LLM and oracle graphs reach none of these out-of-lineage errors. Residual-stale results measure in-lineage stale state, while post-repair probe error measures the reader's subsequent answers.

Dependence placement. The base controlled study includes LOW and HIGH placements of the six incorrect roots. Condition ordering is nearly identical across the two placements (rank correlation approximately 0.99). The depth intervention therefore fixes LOW placement while keeping the underlying episode paired across depth, so the sweep isolates derivation length rather than averaging over source-error placement.

D Public Benchmark Details and Full Grid

MemoryAgentBench Conflict Resolution / FactConsolidation provides timestamped facts and questions. We interpret a later fact with the same subject and relation as a correction of the earlier value. For a multi-hop unit, the question is asked once the pre-correction path is complete, the intermediate hop entity is written back as persistent state, and the final question is probed after later corrections. In the reversed-order control, the values associated with each corrected subject–relation key are reversed while time positions are retained; final gold answers are recomputed from the reversed final memory.

Public repair conditions and provenance. All public results in Figures 3 and 8 use oracle path support. At each write-back, the stored hop-k intermediate is linked to the first k subject–relation keys on the path resolved from the facts available at that time. OFF stores no intermediates, and S0 keeps them after correction. S4 invalidates active intermediates whose recorded support intersects the corrected keys. S4R performs the same invalidation and, whenever an intermediate is invalidated, re-asks the multi-hop question using the updated facts and remaining active memory, then stores the returned intermediates with newly resolved path support. Thus S4 performs closure invalidation, and S4R adds re-derivation and write-back. The controlled testbed uses invalidation plus topological re-derivation for all provenance conditions; this public grid separately evaluates the two operations.

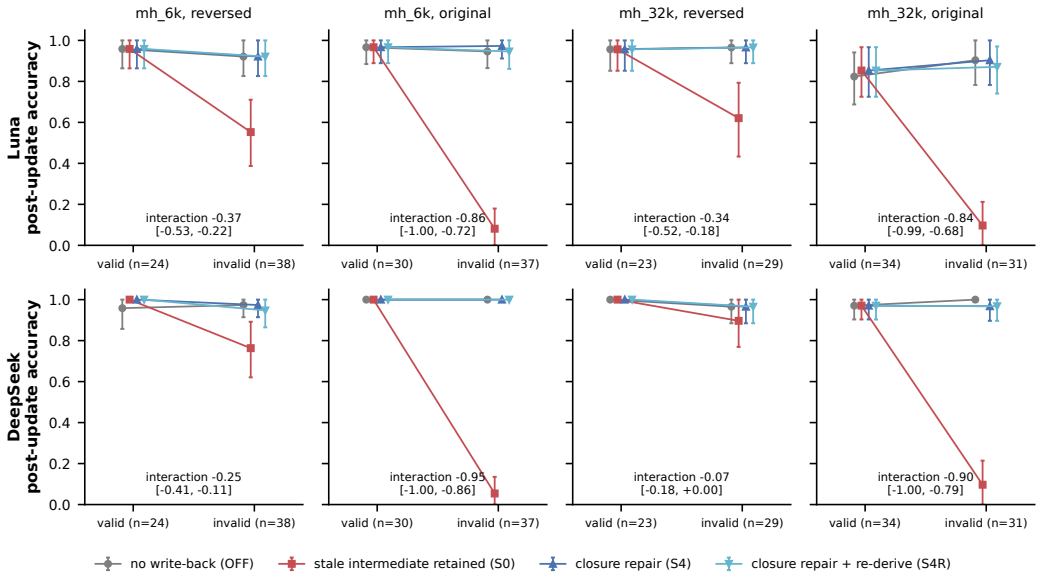

Figure 8: **Full public multi-hop validation grid.** OFF: no write-back; S0: keep intermediates; S4: oracle-path closure invalidation; S4R: S4 plus re-derivation and write-back. Columns show short/long context and original/reversed order; rows show readers. The primary comparison uses reversed short context. Error bars are unit-cluster paired bootstrap 95% CIs.

Table 7: Public write-back × validity interactions. Negative values mean persistence is more harmful when the stored intermediate becomes invalid.

Split	Order	Reader	n valid / invalid	OFF (inv.)	S0 (inv.)	S4 (inv.)	Interaction [95% CI]	$b{:}c$
mh_32k	reversed	Luna	23 / 29	0.966	0.621	0.966	-0.34 [-0.52, -0.18]	10:0
mh_32k	reversed	DeepSeek	23 / 29	0.966	0.897	0.966	-0.07 [-0.18, 0.00]	2:0
mh_32k	original	Luna	34 / 31	0.903	0.097	0.903	-0.84 [-0.99, -0.68]	25:0
mh_32k	original	DeepSeek	34 / 31	1.000	0.097	0.968	-0.90 [-1.00, -0.79]	28:0
mh_6k	reversed	Luna	24 / 38	0.921	0.553	0.921	-0.37 [-0.53, -0.22]	14:0
mh_6k	reversed	DeepSeek	24 / 38	0.974	0.763	0.974	-0.25 [-0.41, -0.11]	8:0
mh_6k	original	Luna	30 / 37	0.946	0.081	0.973	-0.86 [-1.00, -0.72]	32:0
mh_6k	original	DeepSeek	30 / 37	1.000	0.054	1.000	-0.95 [-1.00, -0.86]	35:0

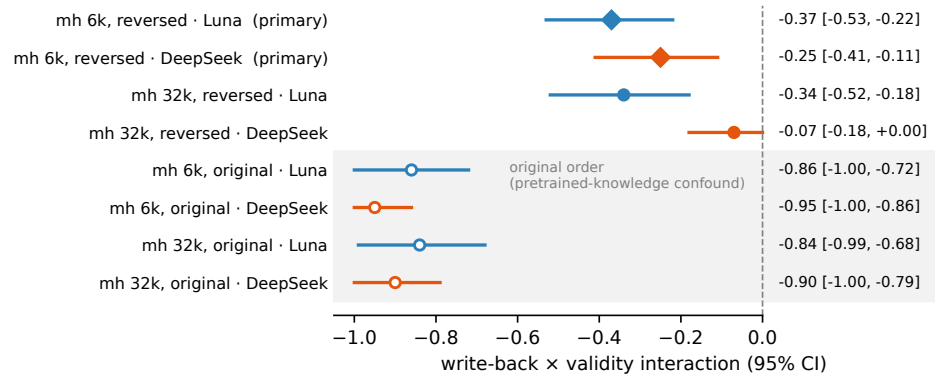

Figure 9: **Write-back × validity interactions across all public cells** (Table 7). Diamonds mark the primary reversed short-context cell; open markers and shading mark original-order cells, which reintroduce the pretrained-world-knowledge asymmetry. Only the longer-context reversed DeepSeek cell has an interval touching zero. Intervals are unit-cluster paired bootstrap 95% CIs.

E Evaluation Counts

Table 8: Evaluation-unit counts used in the reported analyses.

Evaluation	Count
Controlled base episodes	50
Depth variants	50×5
Roots / episode	24
Incorrect roots / episode	6
Base in-stream derived queries / episode	9
Public single-hop units (6k+32k)	137
Public multi-hop 6k, original / reversed	67 / 62
Public multi-hop 32k, original / reversed	65 / 52
Primary reversed valid / invalid units	24 / 38