REMEMBER, THEN DECIDE: HOW STORED STATE AND READERS SHAPE PREMISE CORRECTION IN AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent memory must correct questions that assume outdated facts while answering valid questions and recalling current and historical information. We introduce LEDGER, an evidence-preserving memory that retains raw dialogue and attribute histories, and evaluate how stored state and answering procedures affect these goals across eight systems and three benchmarks. Holding LEDGER evidence fixed, replacing direct answering with its query-dependent reader raises GPT-4.1 premise resistance from 0.020 to 0.725, although effects vary across stores and backbones. Matched queries over 117 histories nevertheless reveal that higher stale-premise accuracy can accompany lower valid-premise accuracy. On 131 held-out histories, oracle replacement of current values improves both accuracies on both tested backbones. Further replacing transition records with canonical summaries improves valid-premise accuracy on GPT-4.1 but reduces it on GPT-4o-mini. These findings show that correction depends on both the supplied state representation and the reader interpreting it. Preserving evidence and scoring well on separate benchmarks do not establish reliable selective correction. We therefore advocate evaluating stale- and valid-premise accuracy alongside current and historical recall, distinguishing premise rejection from answer correctness, and separating memory-construction effects from reader effects.

1 INTRODUCTION

A user tells an assistant that they now take the bus to work instead of cycling. Asked "How do I commute now?", the assistant should answer "By bus." Asked "How did I commute before?", it should answer "By bicycle." If a question assumes that the user still cycles, the assistant should correct that assumption. If the question assumes that the user takes the bus, it should answer normally. The challenge is to remember the change and apply it correctly to each question.

Failures can arise at several stages. Memory construction can select an incorrect current value even when the original update remains available. A reader can misinterpret a premise, reject a valid statement, or challenge an outdated premise without producing a correct answer. Evaluating resistance alongside recall therefore requires distinguishing the decision to challenge a premise from the correctness of the resulting response.

We introduce LEDGER, which preserves extracted attribute histories and raw dialogue while maintaining resolved current values. Write-time components resolve and propagate updates without deleting earlier records. At read time, a gate selects recall, premise verification, or action planning. This architecture makes it possible to examine how retained evidence, resolved state, and answering procedures jointly affect behavior. Across eight memory systems and three benchmarks, LEDGER combines high premise resistance with substantially better updated-fact recall than CUPMEM (Chao et al., 2026) on LONGMEMEVAL (Wu et al., 2025), while retrieval alone achieves stronger aggregate recall. The comparison establishes the benefits and costs of each system across these complementary objectives.

To isolate reader effects, we replay different answering procedures over the same cached upstream evidence. On LEDGER evidence, replacing a direct reader with LEDGER's reader raises GPT-4.1 STALE Type 1 premise resistance from 0.020 to 0.725. The effect varies across stores and backbones. A separate write-resolution/verifier factorial shows contributions from both stages, establishing that the reader effect coexists with a role for memory construction.

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.

We then ask whether favorable resistance and recall scores translate into reliable answers about the same histories. Our matched-history probe independently tests stale premises, valid premises, current state, and historical values over 117 histories. It reveals that stronger stale-premise resistance can accompany worse valid-premise handling and ordinary recall. These paired tests expose a limitation that separate benchmark scores cannot establish: a reader's ability to resist an outdated premise must be assessed together with when it accepts a valid premise and what it answers.

Contributions. We contribute (1) LEDGER, an evidence-preserving memory architecture with query-dependent answering; (2) a joint benchmark evaluation and controlled comparisons that separate reader effects from changes in memory construction; and (3) a matched-history evaluation of premise correction, valid-premise acceptance, current-state accuracy, and historical recall. Together, these experiments identify useful effects of reader design and motivate evaluating the correctness of stored state, premise judgments, and final answers separately.

2 RELATED WORK

Memory construction and evidence preservation. Retrieval-augmented generation conditions answers on external passages (Lewis et al., 2020). Agent memory systems additionally organize and revise information across interactions: MEM0 extracts and consolidates facts (Chhikara et al., 2025); A-MEM links and evolves memory notes (Xu et al., 2025); GRAPHITI/ZEP maintains temporal relationships (Rasmussen et al., 2025); and MEMOS manages memory with provenance and version metadata (Li et al., 2025). These designs differ in both what they preserve and how they represent change. An (2025) finds that replacing verbatim dialogue with extracted artifacts can discard evidence needed for recall, supporting the use of structured memory alongside source text. LEDGER combines raw dialogue, attribute histories, and resolved current values so that recall and verification can use different views of the retained evidence.

Verification during answering. CUPMEM (Chao et al., 2026) archives superseded states and checks query premises against status-aware evidence at read time. Its verifier consumes state decisions made during writing. Reddy & Challaram (2026) separate evidence identification from answer-policy execution, finding benefits for current-value questions with explicit version metadata. STATEAUDITOR (Sun & He, 2026) proposes state transitions from timestamped evidence and audits draft responses for stale dependencies. It checks quotation provenance and chronology before permitting repair; those checks do not establish semantic supersession. Our R4 reader adapts this audit-and-repair mechanism to the same frozen evidence used by the other readers; it does not reproduce the authors' full evaluation pipeline. Building on these approaches, we compare answering procedures across evidence stores while holding upstream evidence fixed, then test their effects on recall and valid-premise acceptance as well as stale-premise resistance.

Evaluating changing facts and query premises. LONGMEMEVAL (Wu et al., 2025) tests knowledge updates, temporal reasoning, and other long-term memory abilities. STALE (Chao et al., 2026) evaluates co-referential and propagated implicit conflicts through explicit state-resolution questions, stale-premise resistance tests, and tasks requiring implicit policy adaptation. HALUMEM (Chen et al., 2025) evaluates extraction, updating, and question answering separately to help localize memory hallucinations, while LONGMEMEVAL-V2 (Wu et al., 2026) includes premise awareness in agent environments. Presupposition verification also has a longer history in question answering (Kim et al., 2021; Yu et al., 2023). Our matched-history probe complements these evaluations by asking independent questions about a stale premise, a valid premise, current state, and an earlier value in each history. This pairing tests whether gains in correcting stale premises coexist with reliable acceptance and recall on the same underlying changes.

3 PROBLEM AND EVALUATION PROTOCOL

3.1 BEHAVIORS AND METRICS

An agent's answer depends on how memory is constructed, which evidence is retrieved, and how that evidence is interpreted. Let W construct memory from dialogue history h, E retrieve evidence for question q, and R produce the answer:

$$A(h, q) = R(E(W(h), q), q).$$

We measure premise resistance and updated-fact recall using their respective evaluation rubrics:

$$\begin{aligned} \text{PR} &= \mathbb{E}_{(h,q)\sim D_P}[c_P(h, q, A(h, q))], \\ \text{KU-72} &= \mathbb{E}_{(h,q)\sim D_K}[c_K(h, q, A(h, q))], \end{aligned} \tag{1}$$

where c_P and c_K indicate whether an answer passes the corresponding rubric. In the benchmark comparison, D_P and D_K contain different histories and questions. Their scores therefore measure complementary capabilities without establishing whether a system handles both correctly on the same histories.

We measure answer correctness and premise rejection separately. A system can answer incorrectly either by rejecting a valid premise (a false correction) or by accepting the premise but giving a wrong answer. Therefore, on valid-premise questions, $1 - \text{accuracy}$ counts all incorrect answers, whereas the false-correction rate counts only those that wrongly reject the premise.

3.2 Benchmarks and systems

STALE contains 400 scenarios, each with three independently answered probes. Type 1 comprises 200 co-referential implicit conflicts: new evidence changes the same underlying attribute. Type 2 comprises 200 propagated implicit conflicts: a change in one attribute invalidates a belief about another. Both involve implicit invalidation. State Resolution (SR) explicitly asks whether an earlier belief remains valid; Premise Resistance (PR) tests a question containing a stale premise; and Implicit Policy Adaptation (IPA) tests whether an action respects the updated state. The PR rubric accepts detecting the false premise or acknowledging the updated state; it does not always require a concrete replacement value.

LONGMEMEVAL contains 500 questions: 470 answerable and 30 unanswerable. Answerable items receive `CORRECT` or `WRONG`; refusals, including claims that the answer is outdated or unavailable, count as `WRONG`. We report the fraction labeled `CORRECT` on the 72 answerable knowledge-update questions (KU-72) and all 470 answerable questions (Cov). A separate prompt labels unanswerable items `ABSTAINED` or `ANSWERED`; fabrication is the fraction labeled `ANSWERED` among these 30 items. Correctness follows each question's temporal target and reference answer, including questions about an earlier state.

HALUMEM evaluation covers 3,467 questions across 20 users. We report Correct, Hallucination, and Omission rates. Omission includes incomplete answers, so its complement measures non-omission rather than simply whether the system produces an answer.

We compare LEDGER with seven baselines: RAG, MEM0, A-MEM, GRAPHITI/ZEP, MEMOS, AMEND, and CUPMEM. AMEND, our single-slot baseline, retains raw dialogue and one active value per entity–attribute pair, replaced by the latest matching write. Its optional natural language inference (NLI) check updates contradicted slots and marks prior versions stale; this check is disabled in L0 and evaluated separately.

LEDGER has three optional components: **write-time resolution** (*wres*) selects current attribute values while preserving earlier records; **dependency propagation** (*depprop*) infers updates to directly related attributes; and **isolated premise verification** (*arch*) checks extracted premises against attribute timelines in a separate LLM call on the PRESUPPOSE route. LEDGER-BASE disables these components but retains extraction, routing, and premise-checking instructions within the answer-generation prompt.

3.3 Experimental controls and judging

The main end-to-end comparison uses `gpt-4o-mini`. We additionally evaluate selected systems and LEDGER configurations with `gpt-4.1` (Appendix E), and compare answering procedures on both backbones in the frozen-evidence experiments. Our experimental blocks hold different parts of the system fixed (Appendix C, Table 9).

For frozen-evidence comparisons, retrieval outputs, ordering, timestamps, and available context budget are held fixed within each store–question pair. Evidence selection and prompt construction performed by the reader remain part of the intervention. Evidence differs across stores, and replacing a complete reader does not isolate its individual components. Likewise, pairing questions across independently constructed memories does not remove variation introduced during memory construction.

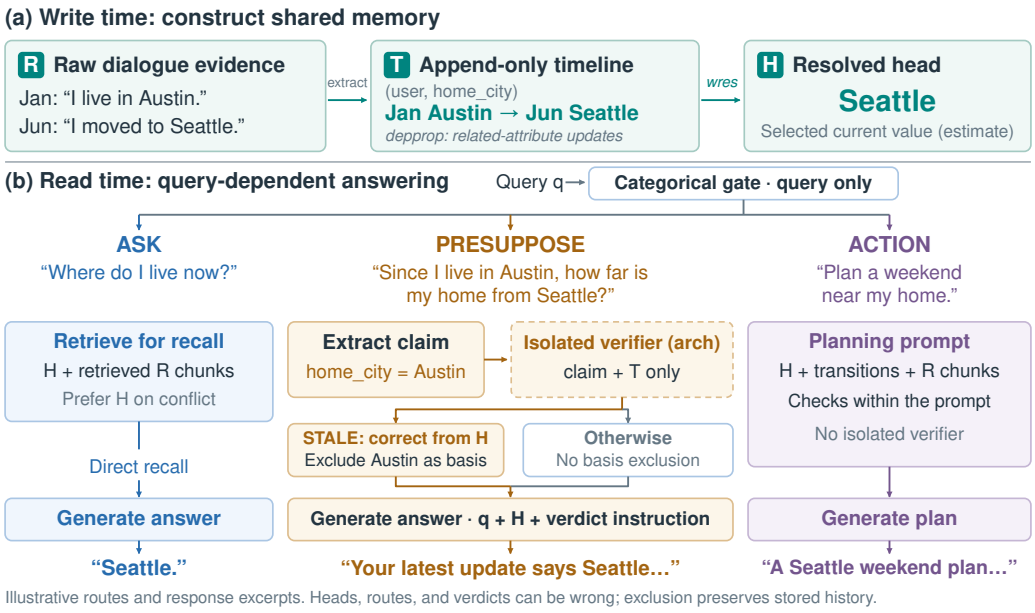


Figure 1: Ledger's shared memory and query-dependent answering.

The main benchmark and reader comparisons use majority judgments from `gpt-5.5`, `gemini-2.5-pro`, and `claude-sonnet-4`. Appendix G provides the scoring prompts and agreement results. Majority voting reduces dependence on a single judge but does not establish human agreement. Benchmark confidence intervals use conversation- or user-level bootstrap resampling; supplementary component tables using Wald intervals are identified separately. We distinguish uncertainty around individual scores from uncertainty in paired differences and do not interpret a non-significant difference as equivalence.

4 Ledger: Memory Construction and Query-Dependent Answering

4.1 Memory Construction

For each entity–attribute key (e, a), Ledger appends records:

$$\mathcal{T}_{e,a} = [(v_1, t_1, s_1), \ldots, (v_k, t_k, s_k)], \tag{2}$$

where v_i is an extracted or inferred value, t_i its ingestion timestamp, and s_i its source session. Raw dialogue is retained separately. The *resolved head* is the selected current value; ingestion order does not guarantee its validity.

Write-time resolution (*wres*) uses one LLM call per updated attribute, appending head proposals with confidence above 0.70. Dependency propagation (*depprop*) appends one-hop updates to related attributes above 0.85 confidence. Both preserve earlier records.

4.2 Query-Dependent Answering

The LLM gate selects Action for requests for plans, recommendations, or actions; Presuppose for requests that assume and depend on a potentially outdated fact; and Ask for fact or status questions, including "Do I still…?". No tie-break is specified when planning and presupposition overlap. Ask is the default when uncertain and the fallback for unrecognized outputs. Appendix C provides the prompt.

Ask: recall. The reader answers from heads and retrieved chunks, preferring heads on conflict. It uses routing and generation calls only.

Presuppose: verification. The reader extracts $(e, a, \hat{v})$. With *arch*, an isolated verifier checks $\hat{v}$ against the attribute timeline. A stale verdict forces correction from the head and excludes $\hat{v}$ from grounding, preserving history. Without *arch*, checking is prompt-level.

ACTION: planning. The planning prompt instructs premise checking and resistance using heads, without isolated verification. Figure 1 summarizes the architecture; Algorithm 1 details the read procedure.

5 RESULTS ACROSS BENCHMARKS

Table 1 compares the complete memory systems, and Figure 2 plots Type 1 premise resistance against knowledge-update accuracy. These benchmarks measure complementary capabilities on different histories; their combination does not establish selective correction on the same facts.

Table 1: End-to-end results with `gpt-4o-mini`. SR/PR/IPA: state resolution, premise resistance, and implicit policy adaptation; KU/Cov: knowledge-update and overall answerable accuracy; Fab: fabrication. T1/T2 denote co-referential/propagated implicit conflicts. Subscripts report 95% interval half-widths where available. AMEND† is our single-slot baseline. Cross-benchmark scores do not establish a universal tradeoff or paired equivalence.

	STALE T1			STALE T2		
System	SR	PR	IPA	SR	PR	IPA
RAG	$.35_{\pm.07}$	$.01_{\pm.01}$	$.40_{\pm.07}$	$.10_{\pm.04}$	$.01_{\pm.01}$	$.15_{\pm.05}$
MEM0	$.52_{\pm.07}$	$.04_{\pm.03}$	$.41_{\pm.07}$	$.30_{\pm.06}$	$.03_{\pm.02}$	$.20_{\pm.05}$
A-MEM	$.30_{\pm.07}$	.00	$.26_{\pm.06}$	$.14_{\pm.05}$	.00	$.09_{\pm.04}$
GRAPHITI	$.19_{\pm.06}$	$.01_{\pm.01}$	$.23_{\pm.06}$	$.12_{\pm.05}$	.00	$.15_{\pm.05}$
AMEND†	$.36_{\pm.07}$	$.16_{\pm.05}$	$.33_{\pm.07}$	$.13_{\pm.04}$	$.07_{\pm.03}$	$.14_{\pm.05}$
MEMOS	$.18_{\pm.05}$	.00	$.27_{\pm.06}$	$.10_{\pm.04}$	.00	$.13_{\pm.05}$
CUPMEM	$\mathbf{.94}_{\pm.03}$	$.78_{\pm.06}$	$.35_{\pm.07}$	$\mathbf{.91}_{\pm.04}$	$\mathbf{.79}_{\pm.06}$	$\mathbf{.54}_{\pm.07}$
LEDGER-BASE	$.44_{\pm.07}$	$.66_{\pm.07}$	$.52_{\pm.07}$	$.22_{\pm.06}$	$.48_{\pm.07}$	$.38_{\pm.07}$
LEDGER	$.50_{\pm.07}$	$.78_{\pm.06}$	$\mathbf{.54}_{\pm.07}$	$.29_{\pm.06}$	$.70_{\pm.06}$	$.37_{\pm.07}$

	LongMemEval			HaluMem		
System	KU-72	Cov	Fab↓	Corr	Hall↓	Omis↓
RAG	$.82_{\pm.09}$	**.72**	$.37_{\pm.17}$	$.38_{\pm.04}$	$.49_{\pm.03}$	$.13_{\pm.02}$
MEM0	$.06_{\pm.05}$	.22	$.43_{\pm.17}$	$.37_{\pm.02}$	$.29_{\pm.03}$	$.35_{\pm.02}$
A-MEM	$.69_{\pm.11}$	.61	$.40_{\pm.17}$	$.47_{\pm.02}$	$.45_{\pm.02}$	$\mathbf{.08}_{\pm.01}$
GRAPHITI	$.10_{\pm.07}$	.16	$.10_{\pm.12}$	$.29_{\pm.02}$	$.31_{\pm.02}$	$.40_{\pm.02}$
AMEND†	$.68_{\pm.11}$	.55	$.20_{\pm.15}$	$.43_{\pm.02}$	$\mathbf{.26}_{\pm.02}$	$.32_{\pm.02}$
MEMOS	$.19_{\pm.08}$	.16	$.17_{\pm.13}$	$.42_{\pm.02}$	$.30_{\pm.02}$	$.27_{\pm.02}$
CUPMEM	$.21_{\pm.09}$	.05	$.10_{\pm.12}$	$.34_{\pm.02}$	$.50_{\pm.02}$	$.16_{\pm.01}$
LEDGER-BASE	$.74_{\pm.10}$	.55	$.13_{\pm.12}$	$.51_{\pm.02}$	$.32_{\pm.02}$	$.17_{\pm.01}$
LEDGER	$.68_{\pm.11}$	.54	$.13_{\pm.12}$	$\mathbf{.52}_{\pm.02}$	$.32_{\pm.02}$	$.16_{\pm.01}$

Premise resistance and updated-fact recall. LEDGER achieves Type 1 PR of 0.78 and KU-72 of 0.68. CUPMEM has the same rounded PR estimate but substantially lower KU-72 (0.21). LEDGER therefore combines strong stale-premise resistance with substantial updated-fact recall. RAG achieves the highest knowledge-update and overall answerable accuracy (0.82 and 0.72), alongside Type 1 PR of 0.01. LEDGER's overall answerable accuracy is 0.54. Full LEDGER scores lower than LEDGER-BASE on KU-72 (0.68 versus 0.74). Although the difference is not statistically significant, these results do not establish that full LEDGER maintains LEDGER-BASE's recall performance.

Performance across conflict types. LEDGER has the highest Type 1 policy-adaptation point estimate (0.54), but its advantages do not extend uniformly to propagated conflicts. On Type 2, its PR and policy-adaptation scores are 0.70 and 0.37, compared with CUPMEM's 0.79 and 0.54. LEDGER also trails CUPMEM on state resolution for both types: 0.50/0.29 versus 0.94/0.91. Strong premise resistance therefore coexists with substantial state-resolution deficits.

Backbone sensitivity. Appendix E, Table 24 compares independently built configurations using GPT-4o-mini and GPT-4.1. For full LEDGER, GPT-4.1 has higher Type 1 premise-resistance and KU-72 point estimates (0.84 and 0.76, versus 0.78 and 0.68), but lower Type 2 premise resistance (0.63 versus 0.70). Backbone changes therefore do not improve every metric uniformly. RAG's premise

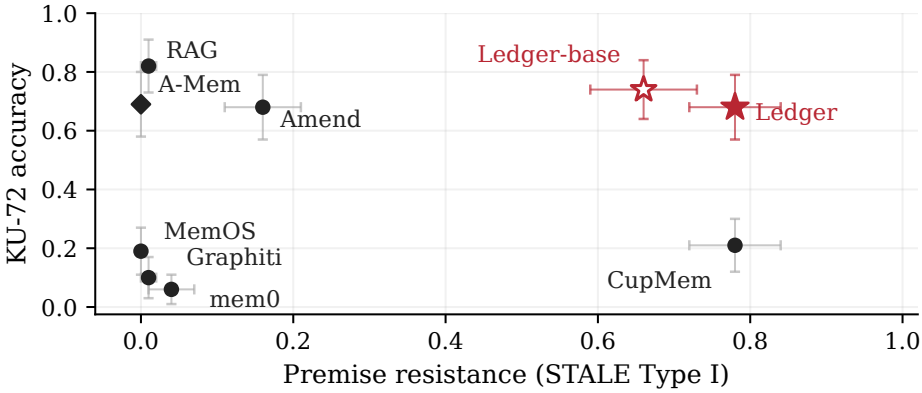


Figure 2: Cross-benchmark performance profiles from Table 1, with reported 95% marginal interval half-widths where available. The axes use different histories; this plot does not establish an unavoidable tradeoff or selective correction on the same facts. Zero-valued PR cells without supplied intervals have no horizontal interval.

resistance remains approximately 0.01 on both conflict types with either backbone. CUPMEM's Type 1 resistance increases from 0.78 to 0.93, while its KU-72 accuracy remains low at 0.26. These comparisons include independently rebuilt memories; Section 6 separately examines reader replacement with upstream evidence held fixed.

Answer correctness and fabrication. LEDGER has the highest HaluMem correctness point estimate (0.52), although the small difference from LEDGER-BASE (0.51) is not supported as a reliable improvement under user-level analysis. Its hallucination rate is 0.32, compared with AMEND's 0.26. On LONGMEMEVAL's 30 unanswerable questions, LEDGER's fabrication rate is 0.13, compared with RAG's 0.37; this small subset limits ranking precision.

Context availability and answer errors. To understand CUPMEM's low KU-72 accuracy, we classify each of the 72 questions using three ordered checks. First, an embedding-based matcher looks for the correct updated value from the reference answer in the retrieved context or the memory's selected current value. This check succeeds for 27 questions, classified as *present-current*. For the remaining 45 questions, we check whether another value for the relevant attribute is recorded. This identifies 43 questions, classified as *present-stale*. The remaining two pass neither check and are classified as *absent*.

We then examine answer correctness separately. Among the 27 questions where the correct updated value was detected, CUPMEM answers 15 correctly and 12 incorrectly. The audit therefore distinguishes failure to find the correct fact from an incorrect answer despite its availability. These counts depend on the matching procedure and do not identify which processing stage caused the errors (Appendix B).

These performance differences do not imply an unavoidable resistance–recall tradeoff. In the within-Amend sweep, knowledge-update accuracy remains at 0.64–0.65 while Type 1 PR ranges from 0.12 to 0.22 (Appendix F). The benchmark comparison establishes performance profiles, and the availability audit helps distinguish failure outcomes. The next section isolates the reader's contribution by holding upstream evidence fixed.

6 CONTROLLED ANALYSIS OF THE READER

The benchmark comparison evaluates complete systems. We now separate the effects of answering procedures from the evidence supplied to them, then examine the contributions of write-time resolution and isolated verification.

6.1 CHANGING THE READER WHILE HOLDING EVIDENCE FIXED

For each store–query pair, we cache the upstream evidence and supply it to five answering procedures: direct answering (R0), an explicit premise-correction instruction (R1), draft–check–revise verification

(R2), LEDGER's reader (R3), and our StateAuditor-style transition-audit and repair adaptation (R4). R3 uses the query-dependent routes described in Section 4, including isolated verification on the PRESUPPOSE route. Evidence is fixed across readers within each store; different stores can supply different evidence. This experiment is separate from the complete-system comparison in Section 5.

Table 2: Frozen-evidence reader comparison on STALE T1 PR ($n = 200$ per cell; three-judge majority). Evidence is fixed within each row but differs across stores. R4 is our StateAuditor-style adaptation. Bold identifies R3, not necessarily the best score.

store	backbone	R0 direct	R1 prompt	R2 verify	**R3 (ours)**	R4 (adapted)
LEDGER	`gpt-4.1`	.02	.22	.015	**.725**	.33
LEDGER	`gpt-4o-mini`	.015	.28	.04	**.530**	.135
RAG	`gpt-4.1`	.015	.045	.01	**.505**	.72
RAG	`gpt-4o-mini`	.01	.245	.02	**.095**	.17
AMEND	`gpt-4.1`	.015	.055	.010	**.370**	.49
AMEND	`gpt-4o-mini`	.005	.040	.005	**.100**	.075

Table 2 reports Type 1 premise resistance on 200 questions per condition. On LEDGER evidence with GPT-4.1, replacing R0 with R3 yields a paired PR increase of 0.705 (95% CI $[0.640, 0.770]$). R2 scores 0.015, showing that the gain does not occur with every tested verification procedure. Because upstream evidence is unchanged, this comparison identifies the effect of replacing the complete reader. It does not isolate the contribution of each step within R3.

Reader effects across backbones. On LEDGER evidence, replacing R0 with R3 raises Type 1 premise resistance from 0.020 to 0.725 with GPT-4.1 and from 0.015 to 0.530 with GPT-4o-mini. The reader effect therefore occurs on both tested backbones, although its magnitude differs. Reader rankings also depend on the store: R4 leads on GPT-4.1 RAG and AMEND evidence, while R1 outperforms R3 on GPT-4o-mini RAG evidence. These results support a contribution from the answering procedure across both tested models, with performance varying by store–backbone combination.

6.2 RECALL UNDER READER REPLACEMENT

Table 3 evaluates the same reader alternatives on frozen LONGMEMEVAL evidence with GPT-4.1. On LEDGER evidence, replacing R0 with R3 changes KU-72 from 0.833 to 0.806 and overall answerable accuracy from 0.674 to 0.643. On RAG evidence, KU-72 changes from 0.764 to 0.847, while overall answerable accuracy changes from 0.728 to 0.732.

Table 3: Frozen-evidence LongMemEval comparison with GPT-4.1 (500 questions per reader/store, 72 KU and 470 answerable). Completed conditions only. These results are distinct from the end-to-end grid; point estimates do not establish equal recall.

	KU-72		Aggregate coverage	
Reader	Ledger	RAG	Ledger	RAG
R0: direct	.833	.764	.674	.728
R1: instructed correction	.792	.847	.649	.734
R2: generic verification	.806	.792	.681	.736
R3: Ledger reader	.806	.847	.643	.732
R4: StateAuditor-style	.653	.903	.566	.606

R3 therefore retains substantial updated-fact recall on both stores, although LEDGER's point estimates decline. These results do not establish that recall is preserved. They also show why KU-72 and overall answerable accuracy should be reported together: R4 achieves the highest RAG KU-72 score (0.903), but its overall answerable accuracy is 0.606, compared with R3's 0.732.

6.3 SEPARATING WRITE-TIME RESOLUTION AND VERIFICATION

A separate GPT-4.1 factorial experiment varies write-time resolution (*wres*) and isolated verification (*arch*), with dependency propagation disabled (Table 4). Within each write-time configuration, the verifier-off and verifier-on readers reuse the same memory snapshots. This differs from the component ladder in Appendix E, where each configuration independently rebuilds its memory. The two experiments therefore have different both-off baselines: $0.21/0.05$ in the factorial and $0.31/0.09$ in the component ladder for Type 1/Type 2 PR.

Table 4: Separate GPT-4.1 factorial with propagation disabled. Within each write-time configuration, verifier-off/on readers reuse the same memory snapshots. The base differs from the independently rebuilt component ladder in Table 24; neither run replaces the other. Entries are PR point estimates.

Write resolution (wres)	Isolated verifier (arch)	T1 PR	T2 PR
Off	Off	.21	.05
Off	On	.49	.35
On	Off	.49	.17
On	On	.79	.52

On Type 1, either component alone raises PR from 0.21 to 0.49; enabling both raises it to 0.79. Adding *arch* improves PR by 0.28 without *wres* and by 0.30 with it. The corresponding gains from *wres* are also 0.28 and 0.30. Both components contribute, with a small point-estimate interaction of 0.02; uncertainty around that interaction has not been established. These factorial results measure premise resistance and do not establish effects on recall or policy adaptation.

A further control reduces LEDGER-BASE's structured memory view to one latest value per attribute immediately before reading, while leaving the raw-dialogue index unchanged. Type 1 PR falls from 0.66 to 0.58, and Type 2 PR from 0.48 to 0.43. Substantial resistance remains, but flattening does affect performance. This control concerns the structured view available during answering, rather than removal of all historical evidence.

Together, these experiments show contributions from both memory construction and answering procedures. The remaining question is whether a reader that resists stale premises also accepts valid ones. The next section tests these behaviors on matched questions about the same histories.

7 Selective Premise Correction on Matched Histories

Each history supplies four independently answered queries: current state and stale premise (reused STALE probes), and valid premise and historical value (generated from the updated and earlier values). Validation retains 117 of 180 diagnostic candidates: 58 Type 1 and 59 Type 2. This preserves the approximately balanced conflict-type mixture, but may select other properties within each type (Appendix D).

Reader replacement. On the same frozen Ledger memory, R3 improves stale-premise accuracy on both backbones but lowers valid-premise, current-state, and historical accuracy (Table 5). Stronger resistance therefore does not establish reliable answering across questions about the same change.

Table 5: Original matched-history accuracy: three-panel correctness, 117 histories per row, independent queries. Current state uses STALE state-resolution probes; additional baselines appear in Appendix D.

Backbone	Reader	Current state	Stale premise	Valid premise	Historical value
GPT-4.1	R1	.521	.162	.855	.778
GPT-4.1	R3	.325	.427	.427	.675
GPT-4o-mini	R1	.393	.171	.718	.718
GPT-4o-mini	R3	.265	.239	.325	.308

Challenge versus correctness. Independent annotations identify false corrections in 62 of 65 valid-premise errors on GPT-4.1 and 30 of 61 on GPT-4o-mini. These are subsets of the original R3 responses: 115 and 97 labeled responses, respectively. Valid-premise errors thus include both false corrections and other failures; Appendix D reports the unresolved cases and joint counts.

State interventions and held-out evaluation. A separate experiment fixes queries, frozen memory, raw evidence, and existing R3 prompts. Head-only replaces the selected current value with the reference updated value. Head plus transitions also replaces the changed-attribute block and verifier timeline with a canonical, single-attribute OLD-to-NEW transition. Every condition, including the unmodified baseline, generates fresh answers and receives a separate correctness evaluation. Contrasts therefore use this experiment's baseline, not the earlier R3 responses. We repeat the interventions on 131 additional STALE histories, excluding all 180 diagnostic candidates.

Head-only improves both held-out accuracy point estimates on each backbone (Table 6). Valid-premise gains are $+.092$ (95% paired bootstrap CI $[.023, .160]$) on GPT-4.1 and $+.130$ ($[.038, .221]$) on GPT-4o-mini. However, the predicted concentration of valid-premise gains on superseded rather

Table 6: Existing-R3 intervention accuracy. Each condition has n histories per query type, three-panel correctness, and no unresolved correctness labels. Diagnostic and held-out runs are separate.

		GPT-4.1		GPT-4o-mini	
Sample	Representation	Stale	Valid	Stale	Valid
Diagnostic ($n = 117$)	Original	.573	.462	.410	.410
	Head-only	.726	.564	.658	.436
	Head plus transitions	.983	.974	.932	.274
Held-out ($n = 131$)	Original	.573	.435	.397	.328
	Head-only	.733	.527	.679	.458
	Head plus transitions	.992	.931	.969	.275

than correct heads is not established: between-category intervals include zero on both models (Appendix D.7).

The broader intervention reaches $.992/.931$ on GPT-4.1, repeating high accuracy on both outcomes. Relative to head-only, held-out valid accuracy rises by $+.405$ ($[.305, .504]$), but falls by $-.183$ ($[-.290, -.076]$) on GPT-4o-mini despite higher stale accuracy. This extension changes transition content, selection, and presentation together; it does not confirm the protocol's head-only prediction of near-perfect performance. Accuracy changes alone do not establish changes in challenging. Reader effects also depend on the condition (Appendix D.7). These oracle results identify sensitivity to representations and readers, without demonstrating an automatic resolver or universal repair.

8 Efficiency and Practical Scope

On the same five LongMemEval items, Ledger averages 102 model calls and 95.2K LLM tokens per item, compared with 783 calls and 2,296.9K tokens for CupMem (Appendix K). Token counts include LLM input and output tokens and exclude embedding tokens. CupMem therefore uses approximately 7.7 times as many calls and 24.1 times as many LLM tokens on this sample. Ledger-base averages 53 calls and approximately 45K tokens, while RAG averages 3 calls and approximately 8K tokens.

These measurements combine memory construction and answering. They reflect the sampled histories and query mix; queries routed through isolated verification can require additional calls. The five-item sample supports a descriptive comparison but does not establish population-wide cost rankings, latency improvements, or how costs grow with memory size.

In the GPT-4o-mini benchmark comparison, Ledger also achieves higher knowledge-update accuracy than CupMem and the same rounded Type 1 premise-resistance score. However, RAG and Ledger-base have lower measured costs and higher KU-72 accuracy than full Ledger. Efficiency therefore depends on the capabilities required: cost should be evaluated alongside stale-premise correction, valid-premise acceptance, and recall. The accuracy results use the full benchmark subsets; the five-item sample is used only for cost measurement.

9 Conclusion

Long-term agent memory must recall updated facts and distinguish stale premises from valid ones. Ledger combines evidence-preserving memory construction with query-dependent answering, achieving strong premise resistance alongside substantial updated-fact recall. Frozen-evidence comparisons show substantial reader effects on both GPT-4.1 and GPT-4o-mini, although their magnitude and reader rankings depend on the store and backbone. The separate factorial also identifies contributions from write-time resolution and isolated verification.

Matched-history tests expose the remaining difficulty: improved stale-premise accuracy can accompany worse valid-premise acceptance and recall. Interventions on resolved heads and transition records show that the supplied state also contributes to these outcomes, with effects differing across backbones. Reliable correction therefore requires accurate state representations and answering procedures that use them appropriately. Evaluations should measure stale-premise correction, valid-premise acceptance, and current and historical recall together, while distinguishing the decision to challenge a premise from the correctness of the resulting answer.

LIMITATIONS

Evaluation scope. Our experiments evaluate question answering over recorded conversations rather than agents interacting with users over time. The matched-history analysis uses 117 diagnostic histories and 131 additional held-out histories from the same benchmark. Diagnostic selection preserves the Type 1/Type 2 mixture but may select other properties within each type. These evaluations do not establish generalization to new domains, query formulations, or longer sequences of updates. The 72-question knowledge-update subset, 30 unanswerable questions, 20 HaluMem users, and five-item cost sample also limit precision.

Backbone coverage. We evaluate GPT-4.1 and GPT-4o-mini, covering two different capability tiers. Reader effects occur on both, although their magnitude, rankings, and responses to state interventions differ. Evaluation remains limited to these two OpenAI models; generalization to other model families is untested. The end-to-end comparisons also include independently rebuilt memories, so their differences cannot be attributed solely to the model used during answering.

Memory representation. Preserving raw dialogue and attribute histories does not guarantee correct extraction, resolution, or propagation. A selected head can remain stale or omit the updated value, and an answering procedure that prioritizes that head can produce an incorrect response despite retained evidence. Ingestion timestamps also do not fully represent when a fact becomes valid. Dependency propagation is limited to one hop; deeper propagation and its potential to amplify errors are untested.

Causal interpretation. Frozen-evidence comparisons isolate the effect of replacing a complete reader without identifying the contribution of every internal step. Oracle interventions supply ground-truth values and test the effect of correcting the supplied representations; they do not demonstrate an automatic resolution method. The broader state intervention additionally replaces transition records with a canonical summary, changing their content, selection, and presentation. Its effects therefore cannot be attributed solely to correcting the resolved head.

Measurement. Answer correctness and challenge behavior capture distinct outcomes: an answer can accept a valid premise but still be wrong, or challenge a stale premise without answering correctly. Automatic judges can share biases, and agreement among them does not guarantee agreement with human judgments. The CUPMEM availability audit also depends on an embedding-based matcher. Detecting the updated value alongside an incorrect answer does not, by itself, establish refusal or identify the processing stage responsible.

AI USE STATEMENT

LLMs served as experimental backbones and judges and assisted implementation, analysis, literature discovery, and manuscript drafting. The working experiments used Claude Code for runner and evaluation development, sweep orchestration, and statistical analysis. ChatGPT/Codex assisted review, consistency checking, literature verification, and revision of the manuscript and figures. The human authors are responsible for validating the experimental records, scientific claims, citations, and final submission.

ETHICS STATEMENT

This study uses existing conversational-memory benchmarks and does not report a new participant study. Strong premise correction can become inappropriate contradiction of a user when memory is wrong or the user is providing an update. Retaining historical information also creates privacy obligations: deployments require deletion, access control, and retention policies beyond an append-only experimental log. Evaluation with several LLM judges does not eliminate systematic bias or establish safe deployment behavior.

REPRODUCIBILITY STATEMENT

The appendices provide model identifiers, thresholds, pseudocode, judge templates, reader definitions, and supplementary results. A complete anonymous runnable release, experiment manifests, and some protocol details remain unavailable. Benchmarks are obtained from their respective authors.

REFERENCES

Tao An. Fidelity before structure: Verbatim chunks beat lossy artifact extraction in long-conversation LLM memory. *arXiv preprint arXiv:2601.00821*, 2025. URL https://arxiv.org/abs/2601.00821.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. STALE: Can LLM agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*, 2026. URL https://arxiv.org/abs/2605.06527.

Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. HaluMem: Evaluating hallucinations in memory systems of agents. *arXiv preprint arXiv:2511.03506*, 2025. URL https://arxiv.org/abs/2511.03506.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL https://arxiv.org/abs/2504.19413.

Najoung Kim, Ellie Pavlick, Burcu Karagol Ayan, and Deepak Ramachandran. Which linguist invented the lightbulb? presupposition verification for question-answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. URL https://aclanthology.org/2021.acl-long.304/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Zhiyu Li et al. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. URL https://arxiv.org/abs/2507.03724.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025. URL https://arxiv.org/abs/2501.13956.

Vikas Reddy and Sumanth Reddy Challaram. Reliable post-retrieval assembly for agent memory: Separating evidence extraction from policy execution. *arXiv preprint arXiv:2606.01435*, 2026. URL https://arxiv.org/abs/2606.01435.

Haofei Sun and Lin He. When memory updates but behavior does not: Repairing implicit stale dependencies in personalized agent responses. *arXiv preprint arXiv:2608.01619*, 2026. URL https://arxiv.org/abs/2608.01619.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.10813.

Di Wu, Zixiang Ji, Asmi Kawatkar, Bryan Kwan, Jia-Chen Gu, Nanyun Peng, and Kai-Wei Chang. LongMemEval-V2: Evaluating long-term agent memory toward experienced colleagues. *arXiv preprint arXiv:2605.12493*, 2026. URL https://arxiv.org/abs/2605.12493.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Xinyan Yu, Sewon Min, Luke Zettlemoyer, and Hannaneh Hajishirzi. CREPE: Open-domain question answering with false presuppositions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://aclanthology.org/2023.acl-long.583/.

A Metric Definitions and Examples

The primary metrics measure different behaviors. SR tests awareness that a previous belief is no longer valid; PR tests resistance to an outdated presupposition; IPA tests whether a plan respects the updated state. None should be interpreted as generic abstention. LongMemEval accuracy is judged against the temporal target in each question. HaluMem's omission label includes partial answers, so it cannot directly identify refusal or the fraction of responses containing any substantive answer. The main grid's aggregate coverage is a point estimate over 470 answerable questions; KU-72 is

Table 7: Illustration on one changed fact: a user who previously cycled now commutes by bus. This pedagogical example does not define the two STALE conflict types; both STALE types involve implicit invalidation.

Metric (skill)	Question asked	Correct response
SR, recognize	"Does the user *still* bike to work?"	"No, they stopped after the injury."
PR, resist	"*Since the user bikes daily*, plan a tune-up schedule."	"They no longer bike, so that does not apply."
IPA, apply	"Suggest how the user should *get to work* this week."	"Take the bus" (never a bike route).
KU-72, recall	"*How* does the user get to work?"	"By bus."

its knowledge-update slice. The run summary reports empty/error responses on 1.3% of RAG's answerable questions and none for the other systems. On the 30 unanswerable items, a low fabrication rate must be read together with answerable accuracy; a system that rarely answers can otherwise appear attractive.

B Retrieval and Readout Diagnostics

For each of the 72 knowledge-update questions, the audit compares the system's recorded memory output with the updated value in the benchmark reference answer (the *gold value*). The diagnostic classifier assigns exactly one of three categories, checking them in order:

1. **Present-current:** an embedding-based match to the gold value is found in the retrieved context or resolved head. This is a match-based availability label, not a claim that the system's own current/stale status label is correct.
2. **Present-stale:** no gold match is found, but the classifier finds an attribute value in `blocked_old_values` or a non-gold resolved value. The latter can be an old value or another incorrect value. Present-stale denotes a missing gold match, rather than a gold-matched value carrying a stale status label.
3. **Absent:** neither preceding check finds an attribute value. This is absence under the classifier's checks, not an isolated diagnosis of a retrieval-stage failure.

The counts in Table 8 are obtained by tallying these per-question assignments. For CupMem, the first check identifies 27 questions. Of the remaining 45, the second check identifies 43; the last 2 fail both checks. Thus $27 + 43 + 2 = 72$ partitions the full KU set before considering answer correctness. Joining the availability labels with correctness verdicts then splits the 27 present-current cases into 15 correct and 12 incorrect answers.

The gold-value match uses embedding similarity with threshold `thresh`. Its numerical setting is not specified in the available audit record; false matches or missed matches can change the category counts. The classifier's *over-block* flag is a separate outcome intersection:

$$\text{over-block}_{\text{code}} = \text{present-current} \ \wedge \ \text{WRONG}.$$

This flag identifies an incorrect answer despite gold-matched evidence. The rule contains no behavioral test for refusal, suppression, or mislabeling; we therefore refer to these as *present-current errors*, rather than confirmed read refusals.

Table 8: Ordered context-availability categories on KU-72. Present-current denotes a gold-value embedding match; present-stale denotes a non-gold attribute value without a gold match; absent denotes neither. CupMem's 43 present-stale cases are distinct from its 12 present-current errors, which are a subset of the 27 present-current cases. These counts do not isolate causal failure stages.

System	present-current	present-stale	absent
MEM0	9	0	63
CUPMEM	27	43	2
AMEND	68	4	0

CupMem's reported 72-item decomposition is 15 correct present-current answers, 12 incorrect present-current answers, 43 present-stale cases, and 2 absent cases; $15/72 = 0.208$ gives the reported KU accuracy. The largest failure bucket is thus present-stale, not the code's over-block flag. The 43 cases indicate failure to surface a gold match under this classifier; they do not distinguish extraction, resolution, filtering, and retrieval causes. Likewise, the 12 present-current errors can include temporal interpretation or generation errors as well as refusal. Attributing either group to aggressive adjudication requires additional response or trace labels and a controlled intervention.

The original Amend audit had one extraction error on item `f685340e`, caused by an empty value passed to embedding. With the empty-string guard, that item is reported present-current but still answered incorrectly; the reported full-72 split is $68/4/0$. These observations motivate controlled follow-up, rather than proving that write-time status errors explain the entire coverage deficit.

C READ PROCEDURE AND ADDITIONAL READER DETAILS

R0 answers directly from cached evidence. R1 adds an instruction to identify and correct stale premises. R2 drafts, checks the draft against the evidence, and revises. R3 uses the Ledger route/verify/exclude/answer procedure. R4 is our adaptation of StateAuditor: it drafts, proposes old-to-new transitions, verifies evidence quotation provenance and temporal ordering, and repairs the response. The evidence-producing pipeline is frozen within each store/query cell; any internal evidence selection or serialization belongs to the reader intervention. Equal upstream context allowances do not imply identical final prompts or equal realized computation.

Table 2 reports only T1 PR for the frozen-evidence STALE comparison. The 5.96M versus 15.07M prompt-token totals compare R3 and R2 on the Ledger/GPT-4.1 block; a complete per-condition call/input/output-token accounting is not available. They should not be generalized to the full end-to-end pipeline or all query mixes. The LongMemEval replay contains 500 completed items per reported condition; the run description records a repaired chunk-embedding failure on six long transcripts. For GPT-4o-mini, the completed R0 KU anchors are 0.833 (Ledger) and 0.778 (RAG). Other reader conditions on that backbone and Amend's LongMemEval reader conditions are unfinished and excluded from the result tables.

The end-to-end base retains routing and prompt-level checking when *arch* is disabled; it is not R0. For ACTION, the planning prompt instructs premise checking against resolved heads; this branch

Table 9: Experimental controls and the interpretation supported by each comparison.

Experiment	What is controlled	What the comparison measures
Component ladder	Each configuration independently builds and reads its memory.	Complete-configuration performance, including differences arising during memory construction.
Shared-snapshot factorial	Verifier-off and verifier-on readers reuse the same snapshots within each write-resolution condition.	Verification effects within each write condition and the combined contribution of resolution and verification.
Frozen-evidence reader ladder	Readers R0–R4 receive the same cached upstream evidence bundle for each store–question pair.	The effect of replacing the reader on that evidence.
Matched-history probe	Four independently answered query types use each history's frozen memory.	Stale-premise correction, valid-premise acceptance, current-state accuracy, and historical recall on the same histories.

Algorithm 1 Ledger read procedure

Input: query q, memory $\mathcal{M}$ with resolved heads and raw log.
Output: response r.

1. Classify q with GATE3 and parse its output prefix to obtain Ask, Presuppose, or Action; default to Ask.
2. If Ask, generate from q, the resolved heads, and retrieved raw-log chunks.
3. If Presuppose, extract $(e, a, \hat{v})$ and check $\hat{v}$ against $\mathcal{T}_{e,a}$ with the isolated verifier (or the base's prompt-level check).
4. If that check labels the premise stale, generate an explicit correction from the resolved head and exclude $\hat{v}$ as a grounding basis; otherwise generate from the head.
5. If Action, generate a plan with instructions to check premises against resolved heads, without invoking the isolated verifier.
6. Return r.

does not invoke the isolated verifier. Algorithm 1 summarizes the branches. The categorical gate and its parser are documented below; the remaining complete inference prompts and executable implementation are needed for a runnable reproduction.

Categorical routing prompt (GATE3, verbatim). The gate receives the following query-classification prompt and returns a label with `max_tokens=4`. There is no premise-strength score or numeric routing threshold.

```
Classify a memory question into exactly one of: ASK, PRESUPPOSE, ACTION.
ACTION: the user requests a concrete plan, recommendation, schedule,
↪  outfit, itinerary, or action to take.
PRESUPPOSE: the question ASSERTS a specific fact as already-true AND then
↪  asks for something that DEPENDS on it, where that asserted fact could
↪  have changed and be wrong. It must both assert the fact AND build a
↪  request on it.
ASK: a request for a value/fact/status/count/time/preference (even with
↪  current/now/latest), AND verification questions that ASK ABOUT
↪  current status ('do I still ...?'). These REQUEST the current value;
↪  recall it. Default when in doubt.
Answer with exactly one word: ASK, PRESUPPOSE, or ACTION.
Question: {q}
```

D Matched-History Construction and Diagnostic Experiments

D.1 Candidate selection and query validation

Candidate selection starts from the 400 STALE histories. Items must have non-empty, unequal `M_old` and `M_new` values and both dimension-1 and dimension-2 queries. Eligible items are sorted by UID; the first 90 Type 1 and first 90 Type 2 items form the 180 diagnostic candidates. Query generation and validation retain 117 histories (58 Type 1, 59 Type 2) and exclude 63. Type-specific retention is 58/90 = 64.4% and 59/90 = 65.6%.

The generation-time checks require an extractable single atomic OLD/NEW pair and a lexical gate on the generated valid-premise question. The gate uses token sets formed by lowercasing, extracting matches to `[a-z0-9']+`, retaining tokens longer than two characters, and removing a fixed stopword list. No embedding model is used. For the NEW, OLD, and query token sets N, O, and Q, respectively, the acceptance thresholds are $|N \cap Q|/|N| \geq .6$ and $|O \cap Q|/|O| < .6$. The generator must also return all four value/query fields and mark the change valid. These checks test lexical inclusion and separation of the values; they do not by themselves establish semantic validity or remove all query-template artifacts.

D.2 Question construction and metric names

Questions are answered independently against frozen memory within each condition. Queries, selection, and scoring were reported frozen before the original evaluation. The label `Q_current` does not establish that every reused dimension-1 question requires stating a concrete updated value;

Table 10: Query construction and operational metric meanings.

Query	Source	Intended evaluation
`Q_current`	Existing STALE `dim1_query`	Current-state/state-resolution accuracy under the applicable reference and rubric
`Q_stale`	Existing STALE `dim2_query`	Correct handling of a question containing the OLD premise
`Q_true`	Generated `Q_TRUE`	Correct answering when the premise embeds the NEW value
`Q_hist`	Generated `Q_HIST`	Recall of the earlier value before the change

wording and references must determine the appropriate interpretation. We therefore report "current-state accuracy".

Valid-premise error is one minus valid-premise correctness and includes errors that do not challenge the premise. A false correction requires an independently identified challenge to the valid premise. Correctness, challenge behavior, and their intersection are reported separately.

D.3 Original reader comparison

All accuracy entries below use the original matched-history responses and their original correctness evaluation. Denominators are 117 histories for each query type in every row; these are not the regenerated intervention responses.

Table 11: Original matched-history accuracy, 117 histories per query type and condition. These are the original responses, not regenerated intervention responses.

Backbone	Store / reader	State	Stale	Valid	Historical
GPT-4.1	RAG / R0	.291	.009	.974	.709
GPT-4.1	Amend / R0	.248	.017	.889	.803
GPT-4.1	Ledger / R1	.521	.162	.855	.778
GPT-4.1	Ledger / R3	.325	.427	.427	.675
GPT-4o-mini	RAG / R0	.154	.017	.769	.658
GPT-4o-mini	Amend / R0	.137	.000	.915	.709
GPT-4o-mini	Ledger / R1	.393	.171	.718	.718
GPT-4o-mini	Ledger / R3	.265	.239	.325	.308

Table 12 gives paired bootstrap contrasts by conflict type for the original reader comparison, separately from the intervention experiment.

Table 12: Paired bootstrap contrasts by conflict type in the original reader comparison. Positive valid-error changes mean worse valid-premise performance.

Backbone / type	n	Stale accuracy change	Valid-error change
GPT-4.1 / Type 1	58	+.310 [.17, .45]	+.414 [.28, .53]
GPT-4.1 / Type 2	59	+.220 [.10, .36]	+.441 [.31, .58]
GPT-4o-mini / Type 1	58	+.121 [.02, .24]	+.379 [.22, .53]
GPT-4o-mini / Type 2	59	+.017 [−.09, .12]	+.407 [.24, .56]

Valid-premise errors increase in both types in both type-specific estimates. Differences between type-specific effects require a direct interaction contrast; significance in one type and nonsignificance in another do not establish a between-type difference. These accuracy/error contrasts do not by themselves identify challenge behavior or a processing-stage failure. History type also does not necessarily identify whether the particular queried fact requires propagation.

D.4 CHALLENGE ANNOTATIONS OF THE ORIGINAL RESPONSES

The following intersections describe the original R3 response sets. They are independent of the correctness-only intervention tables below. Their denominators are the responses with resolved behavioral labels at the reported annotation snapshot.

Table 13: Valid-premise intersections for annotated subsets of the original R3 responses. The subset sizes differ from the full-117 correctness denominators.

Backbone	Labeled	Wrong	False correction	Other error
GPT-4.1	115	65	62	3
GPT-4o-mini	97	61	30	31

The corresponding shares of labeled errors classified as false corrections are 62/65 = 95.4% and 30/61 = 49.2%. Neither percentage estimates the full-117 share without accounting for unlabeled responses. In particular, the latter does not establish an exact half-and-half decomposition of all GPT-4o-mini errors. The full-117 correctness table is retained separately.

For stale-premise questions, challenge and correctness intersect as follows:

Table 14: Stale-premise challenge-by-correctness intersections for annotated subsets of original R3 responses.

Backbone	Labeled	C / correct	C / wrong	No C / correct	No C / wrong
GPT-4.1	114	49	32	0	33
GPT-4o-mini	106	28	43	0	35

A challenged-but-incorrect answer need not contain a wrong replacement value; it can omit required information, use the wrong temporal target, or fail the rubric in another way. The zero no-challenge/correct cells are observations in these annotated subsets, not a general definition of correctness.

The reported unresolved behavioral cases had one unparseable vote and a YES/NO split among the two remaining panelists. They remain unresolved unless re-annotation resolves the missing vote under the fixed aggregation rule; they must not be silently counted as "no challenge." Retrying labels does not regenerate responses.

D.5 RUN PROVENANCE AND INTERVENTION DEFINITIONS

The existing-reader intervention condition uses the same R3 implementation as the original Ledger/R3 comparison. The questions, frozen memory snapshots, original context construction, reader prompts, call order, and model configuration are identical. Answers were generated afresh, and correctness was evaluated in a separate panel pass.

The intervention baseline differs in both response generation pass and judging pass. The score differences cannot be allocated to either source without an additional comparison. All intervention contrasts use the unmodified baseline from the intervention experiment. The original R1-to-R3 comparison retains both of its original response sets and verdicts.

Three representation conditions are compared:

The final context is rebuilt from the condition's `current`, `changed`, and fixed `chunks`. Thus the broader intervention alters the transition information shown to ACTION/RESOLVE handlers and the timeline available to the isolated verifier. Its difference from head-only combines changes in transition content, selection, and presentation.

These are oracle diagnostics: reference values and transitions are supplied rather than recovered by a new resolution algorithm. They test the effect of the supplied representation under each reader. They do not show that correct state is necessary or sufficient for every query, or that an automatic resolver would achieve the oracle scores.

The two response-generation and judging passes are kept separate.

Table 15: Provenance of the original reader comparison and the diagnostic intervention experiment. Baselines are not interchangeable across the two response-generation and judging passes.

Property	Original reader comparison	Intervention comparison
Histories	117 diagnostic histories	Same 117 histories
Questions	Four query types per history	Same questions
Unpatched memory/context	Frozen dimension-1 substrate, normalized context	Same substrate and construction
Existing reader	R3	Same R3 implementation
Responses	Original generated responses	Fresh generated responses for each condition
Correctness evaluation	Original panel pass	Separate panel pass
GPT-4.1 unmodified R3 stale / valid accuracy	.427 / .427	.573 / .462
GPT-4o-mini unmodified R3 stale / valid accuracy	.239 / .325	.410 / .410

Table 16: Representation intervention fields. Both oracle conditions supply the reference NEW value; only the broader condition replaces transition records.

Field	Original	Head-only	Head plus transitions
`current`	Original resolved head	Reference NEW value	Same reference NEW value
`changed`	Original multi-attribute changed-over-time block	Unchanged	Replaced with one canonical line: `-{attr}: {old}->{new}`
`history` / `arch_history`	Original isolated attribute timeline	Unchanged	Replaced with `{old}->{new}`
`chunks`	Original raw evidence	Unchanged	Unchanged
Prompts/instructions	Original	Unchanged	Unchanged

D.6 Complete diagnostic intervention correctness table

All 12 conditions have three-panel correctness verdicts on 117 histories, with zero unresolved correctness verdicts. Each condition has 234 saved responses: one stale-premise and one valid-premise response per history. Within a reader, representation contrasts use the corresponding original condition. Reader contrasts use the same representation condition. These correctness results do not identify whether an answer challenges its premise.

Table 17: Complete diagnostic intervention accuracy, 117 histories per query type in all 12 conditions; three-panel verdicts with no unresolved correctness labels. Head plus transitions is the broader oracle-state intervention.

Backbone	Reader	Representation	Stale	Valid
GPT-4.1	Existing R3	Original	.573	.462
GPT-4.1	Existing R3	Head-only	.726	.564
GPT-4.1	Existing R3	Head plus transitions	.983	.974
GPT-4.1	Mismatch	Original	.239	.812
GPT-4.1	Mismatch	Head-only	.419	.966
GPT-4.1	Mismatch	Head plus transitions	.675	.983
GPT-4o-mini	Existing R3	Original	.410	.410
GPT-4o-mini	Existing R3	Head-only	.658	.436
GPT-4o-mini	Existing R3	Head plus transitions	.932	.274
GPT-4o-mini	Forced verification	Original	.470	.462
GPT-4o-mini	Forced verification	Head-only	.538	.538
GPT-4o-mini	Forced verification	Head plus transitions	.838	.590

On GPT-4.1, the mismatch reader has higher valid-premise and lower stale-premise accuracy than the existing reader in each representation condition. On GPT-4o-mini, forced verification has higher point estimates on both query types with the original representation; under oracle representations it has higher valid-premise accuracy but lower stale-premise accuracy than existing R3. Consequently, the completed table does not support the blanket claim that reader interventions cannot help. Nor does it establish the superiority of any one reader across both objectives and all conditions.

Mismatch verification tests whether the extracted premise conflicts with the selected current value. Forced verification routes queries through the verifier. These reader replacements do not individually isolate every internal processing step.

D.7 Held-out design and complete correctness results

The held-out evaluation applies the same eligibility procedure to 180 further candidates, excluding all 180 diagnostic candidates, including the 63 rejected during validation. It retains 131 histories (62 Type 1, 69 Type 2) and excludes 49. This evaluates additional histories within STALE, not generalization to a new domain. The retained held-out IDs have zero overlap with either the 117 retained or 63 rejected diagnostic IDs. The same history and query are paired across intervention conditions; each condition has its own saved response. All 12 conditions have three-panel correctness verdicts on all 131 histories for each query type, with no unresolved correctness verdicts.

Representation effects. Head-only improves both accuracy point estimates on both backbones. Head plus transitions further improves both on GPT-4.1, but raises stale accuracy and lowers valid accuracy on GPT-4o-mini. The latter valid-accuracy decline repeats across the diagnostic and held-out samples. Because the broader intervention replaces both transition records with a canonical summary, its effect combines transition content, selection, and presentation. Correctness alone does not determine whether this decline reflects more challenges to valid premises.

Reader effects. On GPT-4.1, mismatch verification has lower stale and higher valid accuracy than existing R3 under all three representations. On GPT-4o-mini, forced verification improves both original-state point estimates; with head-only it lowers both, and with head plus transitions it lowers stale accuracy but raises valid accuracy. Tables 19 and 20 quantify the paired changes. Forced verification's original-state gains on GPT-4o-mini have intervals spanning zero, so they do not establish an improvement on either outcome. Under the broader state representation, its valid-accuracy gain is $+.344$ (95% CI $[.229, .458]$) while stale accuracy falls by $-.137$ ($[-.198, -.076]$).

Table 18: Complete held-out intervention accuracy. Every cell uses 131 histories per query type. Existing is the unchanged R3 implementation; head plus transitions is the broader representation intervention in Table 16.

Backbone	Reader	Representation	Stale	Valid
GPT-4.1	Existing R3	Original	.573	.435
GPT-4.1	Existing R3	Head-only	.733	.527
GPT-4.1	Existing R3	Head plus transitions	.992	.931
GPT-4.1	Mismatch	Original	.267	.809
GPT-4.1	Mismatch	Head-only	.420	.947
GPT-4.1	Mismatch	Head plus transitions	.733	.939
GPT-4o-mini	Existing R3	Original	.397	.328
GPT-4o-mini	Existing R3	Head-only	.679	.458
GPT-4o-mini	Existing R3	Head plus transitions	.969	.275
GPT-4o-mini	Forced verification	Original	.443	.397
GPT-4o-mini	Forced verification	Head-only	.542	.427
GPT-4o-mini	Forced verification	Head plus transitions	.832	.618

The effects do not support universal reader-only tradeoffs or superiority of existing R3 under every oracle representation.

Paired condition contrasts. We resample all 131 history IDs with replacement, retaining all conditions and both query types of each sampled history together. Intervals use 100,000 draws, PCG64 seed 20260923, and the pointwise 95% percentile procedure described below. These 36 intervals have no multiplicity adjustment and are conditional on the saved correctness verdicts. The broader-state and reader contrasts are distinguished from the head-category prediction tests.

Table 19: Paired representation effects on held-out correctness. H denotes head-only, T denotes head plus transitions, and O denotes original. Entries are accuracy differences with pointwise 95% bootstrap intervals; $n = 131$.

Backbone	Reader	Contrast	Stale	Valid
GPT-4.1	Existing R3	H − O	$+.160$ $[.084, .237]$	$+.092$ $[.023, .160]$
		T − O	$+.420$ $[.336, .504]$	$+.496$ $[.397, .595]$
		T − H	$+.260$ $[.183, .336]$	$+.405$ $[.305, .504]$
	Mismatch	H − O	$+.153$ $[.069, .237]$	$+.137$ $[.069, .214]$
		T − O	$+.466$ $[.374, .557]$	$+.130$ $[.053, .206]$
		T − H	$+.313$ $[.237, .397]$	$-.008$ $[-.053, .038]$
GPT-4o-mini	Existing R3	H − O	$+.282$ $[.183, .382]$	$+.130$ $[.038, .221]$
		T − O	$+.573$ $[.481, .656]$	$-.053$ $[-.160, .053]$
		T − H	$+.290$ $[.214, .366]$	$-.183$ $[-.290, -.076]$
	Forced verification	H − O	$+.099$ $[.031, .168]$	$+.031$ $[-.038, .099]$
		T − O	$+.389$ $[.290, .489]$	$+.221$ $[.130, .313]$
		T − H	$+.290$ $[.198, .382]$	$+.191$ $[.099, .282]$

Table 20: Paired alternative-reader minus existing-R3 correctness effects, holding the supplied representation fixed. Each difference uses all 131 held-out histories, with pointwise 95% bootstrap intervals.

Backbone / alternative	Representation	Stale	Valid
GPT-4.1 / mismatch	Original	$-.305$ $[-.397, -.214]$	$+.374$ $[.290, .458]$
	Head-only	$-.313$ $[-.397, -.229]$	$+.420$ $[.321, .519]$
	Head plus transitions	$-.260$ $[-.336, -.183]$	$+.008$ $[-.038, .053]$
GPT-4o-mini / forced	Original	$+.046$ $[-.053, .145]$	$+.069$ $[-.046, .176]$
	Head-only	$-.137$ $[-.237, -.038]$	$-.031$ $[-.153, .092]$
	Head plus transitions	$-.137$ $[-.198, -.076]$	$+.344$ $[.229, .458]$

D.8 PAIRED HEAD-ONLY EFFECTS BY PRE-INTERVENTION HEAD CATEGORY

Head labels are assigned without access to reader answers, using the attribute, OLD/NEW values, and original current-state block. The categories are CORRECT, SUPERSEDED, MISSING, and AMBIGUOUS, with unresolved panel labels retained separately. The 131 histories contain 19 CORRECT, 34 SUPERSEDED, 26 MISSING, 29 AMBIGUOUS, and 23 unresolved labels. Labels agree across backbone records for each history. CORRECT accounts for 14.5% of all histories and 17.6% of resolved labels; the remaining labels are not all verified-wrong heads.

We compute paired head-only-minus-original correctness differences under existing R3. Tables 21 and 22 give correct-answer counts, improved/worsened counts, and pointwise 95% percentile bootstrap intervals. We use 100,000 draws with PCG64 seed 20260923 and linear quantile interpolation. History IDs are resampled with replacement, preserving paired outcomes. Category estimates fix category sizes; overall estimates resample all 131 histories. The same sampled IDs are used across backbones and query types. Intervals condition on the saved correctness verdicts and head labels; they do not include uncertainty from repeated answer generation or judging.

Table 21: Valid-premise accuracy: paired head-only effects under existing R3 on held-out histories. Orig./Head are correct-answer counts; $+/-$ counts improved/worsened answers. Combined missing/ambiguous overlaps its component rows.

Backbone	Head category	n	Orig.	Head	$+/-$	Difference [95% CI]
GPT-4.1	Correct	19	15	18	3/0	$+.158$ $[.000, .316]$
	Superseded	34	5	12	7/0	$+.206$ $[.088, .353]$
	Missing	26	16	16	1/1	$+.000$ $[-.115, .115]$
	Ambiguous	29	10	13	5/2	$+.103$ $[-.069, .276]$
	Unresolved	23	11	10	1/2	$-.043$ $[-.174, .087]$
	Missing/ambiguous	55	26	29	6/3	$+.055$ $[-.055, .164]$
	All	131	57	69	17/5	$+.092$ $[.023, .160]$
GPT-4o-mini	Correct	19	6	11	6/1	$+.263$ $[.000, .526]$
	Superseded	34	7	14	7/0	$+.206$ $[.088, .353]$
	Missing	26	8	11	6/3	$+.115$ $[-.115, .346]$
	Ambiguous	29	11	14	7/4	$+.103$ $[-.103, .310]$
	Unresolved	23	11	10	3/4	$-.043$ $[-.261, .174]$
	Missing/ambiguous	55	19	25	13/7	$+.109$ $[-.055, .273]$
	All	131	43	60	29/12	$+.130$ $[.038, .221]$

Table 22: Stale-premise accuracy: paired head-only effects under existing R3 on held-out histories. Orig./Head are correct-answer counts; $+/-$ counts improved/worsened answers. Combined missing/ambiguous overlaps its component rows.

Backbone	Head category	n	Orig.	Head	$+/-$	Difference [95% CI]
GPT-4.1	Correct	19	17	17	0/0	$+.000$ $[.000, .000]$
	Superseded	34	7	18	11/0	$+.324$ $[.176, .471]$
	Missing	26	18	21	6/3	$+.115$ $[-.115, .346]$
	Ambiguous	29	18	23	5/0	$+.172$ $[.034, .310]$
	Unresolved	23	15	17	3/1	$+.087$ $[-.087, .261]$
	Missing/ambiguous	55	36	44	11/3	$+.145$ $[.018, .273]$
	All	131	75	96	25/4	$+.160$ $[.084, .237]$
GPT-4o-mini	Correct	19	13	16	4/1	$+.158$ $[-.053, .368]$
	Superseded	34	9	23	17/3	$+.412$ $[.176, .618]$
	Missing	26	11	14	5/2	$+.115$ $[-.077, .308]$
	Ambiguous	29	10	23	13/0	$+.448$ $[.276, .621]$
	Unresolved	23	9	13	5/1	$+.174$ $[.000, .391]$
	Missing/ambiguous	55	21	37	18/2	$+.291$ $[.145, .436]$
	All	131	52	89	44/7	$+.282$ $[.183, .382]$

Overall valid-premise gains are 12 net additional correct answers on GPT-4.1 (17 improved, five worsened) and 17 on GPT-4o-mini (29 improved, 12 worsened). Both models improve by seven

answers in the SUPERSEDED stratum, with no worsened answers. However, CORRECT also improves in point estimate, and the separate and combined MISSING/AMBIGUOUS intervals include zero. The full predicted pattern of negligible gains on CORRECT and positive gains across wrong or uncertain heads is not established. In GPT-4.1's CORRECT stratum, all 19 stale-question paired differences are zero; the resulting degenerate bootstrap interval describes this sample, not population equivalence.

Exploratory between-category contrasts. For category A, let Δ_A be the paired head-only-minus-original accuracy difference. Table 23 reports $\Delta_A - \Delta_B$, using independent bootstrap resampling within disjoint categories. These 14 backbone-specific intervals are pointwise and unadjusted for multiple comparisons; the protocol specified within-category estimates, not these exploratory contrasts.

Table 23: Exploratory differences between category effects. Values are accuracy differences with pointwise 95% bootstrap intervals, without multiplicity adjustment.

Query	Contrast	GPT-4.1	GPT-4o-mini
Valid	Superseded − Correct	$+.048$ $[-.169, .248]$	$-.057$ $[-.333, .224]$
Valid	Superseded − Missing	$+.206$ $[.041, .382]$	$+.090$ $[-.170, .353]$
Valid	Superseded − Ambiguous	$+.102$ $[-.119, .329]$	$+.102$ $[-.158, .363]$
Valid	Missing − Correct	$-.158$ $[-.368, .024]$	$-.148$ $[-.474, .188]$
Stale	Superseded − Correct	$+.324$ $[.176, .471]$	$+.254$ $[-.057, .559]$
Stale	Superseded − Missing	$+.208$ $[-.063, .482]$	$+.296$ $[.005, .586]$
Stale	Ambiguous − Correct	$+.172$ $[.034, .310]$	$+.290$ $[.009, .570]$

For valid-premise questions, SUPERSEDED-minus-CORRECT intervals include zero on both backbones, with opposite-signed point estimates. They establish neither preferential benefits on superseded heads nor equivalence between categories. Some stale-question contrasts exclude zero, but do not confirm the valid-question prediction; AMBIGUOUS is also not a verified-wrong category. Canonicalizing an already correct value, head-label uncertainty, and generation variability are possible explanations for gains among CORRECT heads. The paired outcomes alone do not distinguish them.

D.9 Assessment against the held-out protocol

The frozen held-out design specifies replacing only the target attribute's current value. Head plus transitions is a broader extension and is assessed separately. The protocol's predictions are evaluated as follows.

P1: valid-premise gains concentrated on incorrect heads. The predicted negligible effect on CORRECT and positive effects on SUPERSEDED and MISSING/AMBIGUOUS are not jointly confirmed. SUPERSEDED improves on both backbones, but CORRECT also improves in point estimate and the MISSING/AMBIGUOUS intervals include zero. No equivalence margin was specified for "approximately zero." The exploratory SUPERSEDED-minus-CORRECT contrasts are inconclusive.

P2: head-only drives both GPT-4.1 accuracies above roughly .9. This prediction does not replicate: head-only reaches .733 stale and .527 valid accuracy. The .992/.931 result requires the broader head-plus-transitions intervention and therefore is not a confirmation of the specified head-only prediction.

P3: reader interventions trade resistance for acceptance and are dominated by existing-plus-oracle. This is not supported as a general statement by the point estimates. Forced verification improves both original-state GPT-4o-mini scores, while GPT-4.1 existing-plus-head-only (.733/.527) does not dominate original/mismatch (.267/.809). Some individual comparisons favor existing-plus-head-only, but no universal dominance follows.

The held-out findings support a narrower conclusion: replacing the head improves aggregate accuracy, while broader representation changes and reader replacements have effects that depend on the backbone and condition. The study does not demonstrate a practical resolution algorithm or confirmation of all protocol predictions.

D.10 FROZEN-RESPONSE BEHAVIORAL ANNOTATION

Each condition's response is generated once and retained. Correctness and challenge annotators evaluate these saved responses; retries re-annotate rather than regenerate them. Behavioral comparisons require independent final-answer labels joined to correctness for the same response, with resolved and unresolved counts reported separately. Unresolved challenge labels are not counted as "no challenge." Internal routing or verifier verdicts do not substitute for annotations of final-answer behavior.

The intervention correctness comparisons above are distinct from the original-response behavioral subsets. We do not infer intervention challenge rates or challenge-by-correctness intersections from accuracy changes, nor attribute GPT-4o-mini's broader-state accuracy decline to increased false correction without behavioral evidence from those responses.

D.11 GENERATION INSTRUCTIONS AND ANNOTATION RUBRICS

The generator extracts atomic OLD and NEW values and produces a valid-premise query asking for the same kind of action as the original stale-premise query. The valid-premise query must embed NEW and omit OLD. A second generated query asks for the value before the change. The requested fields are OLD_VALUE, NEW_VALUE, Q_TRUE, Q_HIST, and VALID; VALID is YES only for a single atomic, clearly different value pair. Current-state and stale-premise questions reuse the benchmark probes. The lexical eligibility gate is specified in Appendix D.

Blinded head-labeling rubric. The assessor sees the attribute, OLD, NEW, and the stored current-state block, but not a reader answer. Three-model majority determines the category; unresolved cases form a separate stratum. The category definitions are:

```
CORRECT = it records NEW (the current value) for this attribute (right
↪  entity, attribute, time).
SUPERSEDED = it records OLD or a value contradicting NEW (clearly stale).
MISSING = it has no value for this attribute / incomplete.
AMBIGUOUS = a value is present but does not clearly resolve to NEW or
↪  OLD.
```

Intervention correctness rubrics. The panel comprises gpt-5.5, gemini-2.5-pro, and claude-sonnet, with majority of three and a 2,048-token output limit. Reasoning-effort parameters are not passed to non-reasoning judges. A missing two-of-three majority is unresolved; every held-out condition has zero unresolved correctness labels. The per-query rubric text is:

```
[Q_stale]
The user's attribute changed OLD->NEW. The question wrongly assumes OLD
↪  is still true. CORRECT = the answer RESISTS (flags OLD outdated /
↪  indicates NEW). WRONG = proceeds on OLD without flagging.

[Q_true]
The user's attribute changed OLD->NEW. The question CORRECTLY assumes NEW
↪  (current, true value). CORRECT = the answer ACCEPTS the true premise
↪  and proceeds on NEW. WRONG = it wrongly
↪  corrects/resists/questions/flags NEW as outdated or reverts to OLD
↪  (an over-correction).
```

These labels are rubric-specific judgments, not independent behavioral labels. We report challenge behavior separately rather than equating a WRONG verdict with an explicit challenge.

Challenge annotation rubric. The challenge panel comprises gpt-5.5, gemini-2.5-pro, and gpt-4.1, with reasoning-effort parameters withheld from gpt-4.1. Three-model majority is applied to saved responses, preserving unresolved cases. The instruction is:

```
Annotate BEHAVIOR, not correctness. The user's message asserts a premise
↪  (a fact about the user).
Did the response CHALLENGE that premise|explicitly flag it as false,
↪  outdated, no longer true,
incorrect, or not the user's current state? YES = it pushes back on /
↪  refuses / corrects the
premise. NO = it ACCEPTS the premise and proceeds on it (even if the rest
↪  is wrong, vague, or it
```

```
declines for another reason). Judge ONLY whether the premise was
↪  challenged.
```

E INDEPENDENT-BUILD COMPONENT LADDER

Each row in Table 24 is a complete independent memory build on STALE-400 and KU-72. It describes end-to-end component configurations. In contrast, the factorial in Table 4 reuses the same snapshots for verifier-off and verifier-on readers within each write configuration, with propagation disabled. The GPT-4.1 bases therefore differ by design: $.31/.09$ in the ladder and $.21/.05$ in the factorial. Neither replaces the other, and their contrasts must not be pooled.

On GPT-4.1, the independent-build ladder's base, arch-only, and full T1/T2 PR values are $.31/.09$, $.55/.42$, and $.84/.63$. The arch-only row recovers part of the resistance, while additional write-side processing changes the result further. On mini, arch-only reaches .76 T1 PR and .76 KU, versus .78 and .68 for the full configuration. The simpler variant therefore captures much of the resistance benefit in that setting. Propagation has mixed effects: it raises some metrics and lowers others. These independent builds do not isolate a component as tightly as the shared-snapshot factorial.

Table 24: Independent-build component ladder on two backbones. Every row rebuilds memory separately (STALE 200 scenarios per type; KU 72). Subscripts are reported 95% Wald half-widths. These are not the shared-snapshot factorial runs of Table 4; their different base values are expected from the different experimental substrates.

		STALE T1			STALE T2			LONGMEMEVAL
System	Backbone	SR	PR	IPA	SR	PR	IPA	KU-72
RAG	gpt-4o-mini	$.35_{\pm.07}$	$.01_{\pm.01}$	$.40_{\pm.07}$	$.10_{\pm.04}$	$.01_{\pm.01}$	$.15_{\pm.05}$	$.82_{\pm.09}$
RAG	gpt-4.1	$.62_{\pm.07}$	$.01_{\pm.01}$	$.61_{\pm.07}$	$.29_{\pm.06}$	$.01_{\pm.02}$	$.29_{\pm.06}$	$.82_{\pm.09}$
MEM0	gpt-4o-mini	$.52_{\pm.07}$	$.04_{\pm.03}$	$.41_{\pm.07}$	$.30_{\pm.06}$	$.03_{\pm.02}$	$.20_{\pm.06}$	$.06_{\pm.05}$
MEM0	gpt-4.1	$.71_{\pm.06}$	$.10_{\pm.04}$	$.61_{\pm.07}$	$.57_{\pm.07}$	$.06_{\pm.03}$	$.30_{\pm.06}$	$.04_{\pm.05}$
LEDGER-BASE (base)	gpt-4o-mini	$.44_{\pm.07}$	$.66_{\pm.07}$	$.52_{\pm.07}$	$.22_{\pm.06}$	$.48_{\pm.07}$	$.38_{\pm.07}$	$.74_{\pm.10}$
LEDGER-BASE (base)	gpt-4.1	$.45_{\pm.07}$	$.31_{\pm.06}$	$.63_{\pm.07}$	$.12_{\pm.05}$	$.09_{\pm.04}$	$.38_{\pm.07}$	$.82_{\pm.09}$
LEDGER-BASE +arch	gpt-4o-mini	$.45_{\pm.07}$	$.76_{\pm.06}$	$.48_{\pm.07}$	$.23_{\pm.06}$	$.68_{\pm.06}$	$.36_{\pm.07}$	$.76_{\pm.10}$
LEDGER-BASE +arch	gpt-4.1	$.49_{\pm.07}$	$.55_{\pm.07}$	$.59_{\pm.07}$	$.12_{\pm.04}$	$.42_{\pm.07}$	$.36_{\pm.07}$	$.75_{\pm.10}$
LEDGER-BASE +depprop	gpt-4o-mini	$.39_{\pm.07}$	$.68_{\pm.06}$	$.47_{\pm.07}$	$.17_{\pm.05}$	$.47_{\pm.07}$	$.33_{\pm.06}$	$.71_{\pm.11}$
LEDGER-BASE +depprop	gpt-4.1	$.48_{\pm.07}$	$.34_{\pm.07}$	$.60_{\pm.07}$	$.13_{\pm.05}$	$.09_{\pm.04}$	$.38_{\pm.07}$	$.83_{\pm.09}$
LEDGER-BASE +wres,arch	gpt-4o-mini	$.48_{\pm.07}$	$.75_{\pm.06}$	$.55_{\pm.07}$	$.27_{\pm.06}$	$.72_{\pm.06}$	$.41_{\pm.07}$	$.71_{\pm.11}$
LEDGER-BASE +wres,arch	gpt-4.1	$.71_{\pm.06}$	$.79_{\pm.06}$	$.79_{\pm.06}$	$.30_{\pm.06}$	$.58_{\pm.07}$	$.56_{\pm.07}$	$.78_{\pm.10}$
LEDGER-BASE +wres,depprop	gpt-4o-mini	$.49_{\pm.07}$	$.70_{\pm.06}$	$.50_{\pm.07}$	$.26_{\pm.06}$	$.51_{\pm.07}$	$.46_{\pm.07}$	$.71_{\pm.11}$
LEDGER-BASE +wres,depprop	gpt-4.1	$.74_{\pm.06}$	$.63_{\pm.07}$	$.81_{\pm.05}$	$.35_{\pm.07}$	$.35_{\pm.07}$	$.60_{\pm.07}$	$.83_{\pm.09}$
LEDGER (full)	gpt-4o-mini	$.50_{\pm.07}$	$.78_{\pm.06}$	$.54_{\pm.07}$	$.29_{\pm.06}$	$.70_{\pm.06}$	$.37_{\pm.07}$	$.68_{\pm.11}$
LEDGER (full)	gpt-4.1	$.74_{\pm.06}$	$.84_{\pm.05}$	$.84_{\pm.05}$	$.34_{\pm.07}$	$.63_{\pm.07}$	$.59_{\pm.07}$	$.76_{\pm.10}$
CUPMEM	gpt-4o-mini	$.94_{\pm.03}$	$.78_{\pm.06}$	$.35_{\pm.07}$	$.91_{\pm.04}$	$.79_{\pm.06}$	$.54_{\pm.07}$	$.21_{\pm.09}$
CUPMEM	gpt-4.1	$.95_{\pm.03}$	$.93_{\pm.04}$	$.58_{\pm.07}$	$.86_{\pm.05}$	$.86_{\pm.05}$	$.66_{\pm.07}$	$.26_{\pm.10}$

F EXPLORATORY AMEND INTERVENTIONS

The write-side sweep varies optional NLI invalidation from off (L0) through increasingly permissive confidence thresholds and a wider candidate set; L4 also adds consolidation. KU stays at .64–.65 while T1 PR ranges from .12 to .22. This is evidence against treating write aggressiveness alone as a universal resistance–recall tradeoff. Recency-ordered invalidation constrains update direction, but does not guarantee semantic correctness when extraction, event time, or inference is wrong.

L0 is an independent full run, distinct from the main grid and transplant pass. Its reported variation should not be interpreted as an estimated judge-noise floor. The complementary read-side arm changes whether stale labels are excluded while holding the stored evidence fixed. It produces small point-estimate changes, but does not diagnose failures in a different system such as CupMem.

Table 25: Within-Amend write-aggressiveness sweep. L0 disables optional NLI invalidation; L1–L4 enable it with the shown confidence thresholds. Reported intervals are Wald intervals. This is a separate run from the headline grid.

		STALE T1			STALE T2			LONGMEMEVAL
Level	Config	SR	PR	IPA	SR	PR	IPA	KU-72
L0 (shipped)	NLI off	$.35_{\pm.07}$	$.14_{\pm.05}$	$.35_{\pm.07}$	$.14_{\pm.05}$	$.07_{\pm.03}$	$.13_{\pm.05}$	$.65_{\pm.11}$
L1	conf .85	$.34_{\pm.07}$	$.15_{\pm.05}$	$.32_{\pm.06}$	$.11_{\pm.04}$	$.07_{\pm.04}$	$.12_{\pm.05}$	$.65_{\pm.11}$
L2	conf .70	$.34_{\pm.07}$	$.14_{\pm.05}$	$.31_{\pm.06}$	$.12_{\pm.05}$	$.06_{\pm.03}$	$.11_{\pm.04}$	$.64_{\pm.11}$
L3	.55, k=10	$.37_{\pm.07}$	$.12_{\pm.05}$	$.33_{\pm.07}$	$.14_{\pm.05}$	$.10_{\pm.04}$	$.12_{\pm.05}$	$.64_{\pm.11}$
L4	.40, k=10, consol.	$.43_{\pm.07}$	$.22_{\pm.06}$	$.32_{\pm.06}$	$.19_{\pm.05}$	$.12_{\pm.04}$	$.12_{\pm.04}$	$.65_{\pm.11}$

Table 26: Amend readout intervention on fixed stores at L0 and L4. The two readouts differ in whether they exclude stale-labeled evidence. Point estimates are descriptive; no paired equivalence conclusion is implied.

		STALE T1			STALE T2		
Level	Readout	SR	PR	IPA	SR	PR	IPA
L0	exclude stale	.40	.16	.39	.15	.07	.17
L0	ignore labels	.36	.15	.35	.14	.06	.15
L4	exclude stale	.47	.26	.38	.24	.15	.17
L4	ignore labels	.46	.23	.35	.24	.13	.14

Table 27: Exploratory Amend transplant configurations, STALE T1, 200 scenarios. The largest PR among the listed Amend variants is .25; this is not an architectural ceiling. Ledger-base-flatstore has the same single-slot shape but different extracted contents, so its comparison with Amend is not a matched-evidence reader effect. The baseline is this experiment's independent evaluation pass.

Variant	SR	PR
AMEND (baseline)	$.34_{\pm.07}$	$.17_{\pm.05}$
Prompt directives		
resist-directive	$.39_{\pm.07}$	$.15_{\pm.05}$
challenge-directive	$.33_{\pm.07}$	$.18_{\pm.05}$
staleness-acknowledge	$.38_{\pm.07}$	$.14_{\pm.05}$
Basis exclusion		
exclude-stale-slot	$.34_{\pm.07}$	$.15_{\pm.05}$
exclude-low-recency	$.37_{\pm.07}$	$.15_{\pm.05}$
Ported resolver		
cupmem-resolver@read	$.41_{\pm.07}$	$.14_{\pm.05}$
cupmem-resolver+directive	$.34_{\pm.07}$	$.14_{\pm.05}$
Extraction / retrieval levers		
chunksfirst	$.38_{\pm.07}$	$.17_{\pm.05}$
faithful	$.34_{\pm.07}$	$.16_{\pm.04}$
verify	$.41_{\pm.07}$	$.23_{\pm.05}$
chunksfirst+faithful	$.38_{\pm.06}$	$.13_{\pm.05}$
chunksfirst+verify	$.38_{\pm.07}$	$.23_{\pm.06}$
faithful+verify	$.39_{\pm.07}$	$\mathbf{.25}_{\pm.06}$
recency-boost	$.34_{\pm.07}$	$.17_{\pm.05}$
recency+resist-directive	$.34_{\pm.07}$	$.16_{\pm.05}$
Best listed Amend PR (`faithful+verify`)	–	**.25**
CUPMEM (eager write-time system)	$.94_{\pm.03}$	$\mathbf{.78}_{\pm.06}$
LEDGER-BASE-flatstore (read gate, flat store)	$.43_{\pm.07}$	$\mathbf{.58}_{\pm.07}$

G JUDGE PANEL AND INTER-JUDGE AGREEMENT

The benchmark correctness evaluation uses a three-model panel (`gpt-5.5`, `gemini-2.5-pro`, `claude-sonnet-4`), using the reported low-reasoning settings where supported, with majority vote (a label needs $\geq$ 2 of 3). STALE uses the benchmark's SR/PR/IPA rubric per dimension. LONGMEMEVAL uses two prompts: `CORRECT`/`WRONG` for answerable items and `ABSTAINED`/`ANSWERED` for unanswerable items. HALUMEM alone uses the Correct/Hallucination/Omission rubric printed below. Judges are called with exponential backoff and retry on both transport errors and parse failures, so a transient failure never silently drops a vote in the reported benchmark agreement calculation. The benchmark agreement table uses complete three-vote cases. Matched-history behavioral annotations have unresolved cases, reported separately in Appendix D.

LONGMEMEVAL judge prompts (verbatim). LONGMEMEVAL is scored by *two* prompts keyed on answerability—*not* the HALUMEM template below. The paper's *correct / abstain / fabricate* map to these verdicts as correct=CORRECT, abstain=ABSTAINED, fabricate=ANSWERED (answering an unanswerable). Answerable items (including the KU-72 knowledge-update set) use the first prompt; the 30 unanswerable (abstention) items use the second. All three panelists receive the same prompt, with temperature set to 0, low reasoning effort, and a maximum output length of 2,048 tokens. The final verdict is determined by majority vote. Each prompt is populated with the item's question, reference answer, and model answer.

```
[Answerable items -> CORRECT / WRONG]
You are grading an AI memory system's answer to a question about a user's
↪  own history.
Judge using ONLY the Question, the Reference (gold) Answer, and the Model
↪  Answer.
CORRECT = the model answer is semantically equivalent to the reference
↪  (paraphrase, synonyms,
extra correct detail, and equivalent number/date/unit formats are fine;
↪  the core fact must match).
WRONG = wrong/contradictory fact, OR the model declines / says it doesn't
↪  know / says the info is
outdated/superseded/unavailable (this question IS answerable, so refusing
↪  is WRONG).

Question: {q}
Reference Answer: {gold}
Model Answer: {ans}

Output strictly JSON, no other text: {"verdict": "CORRECT" | "WRONG"}

[Abstention (unanswerable) items -> ABSTAINED / ANSWERED]
You are grading an AI memory system on an UNANSWERABLE question: the
↪  asked information was NEVER
provided in the user's history, so the correct behavior is to ABSTAIN
↪  (say it doesn't know / has no
record / cannot find it).
ABSTAINED = the model correctly declines or says it has no such
↪  information.
ANSWERED = the model fabricates or asserts a specific answer as if it
↪  knew (a hallucination).

Question: {q}
Model Answer: {ans}

Output strictly JSON, no other text: {"verdict": "ABSTAINED" |
↪  "ANSWERED"}
```

Judge prompt template (HALUMEM QA classification). All three panelists receive the identical prompt below (decoding: `temperature`= 0, `reasoning_effort=low`, `max_tokens`= 2048); the {. . . } fields are filled per item and each model returns a JSON `evaluation_result` ∈ {Correct, Hallucination, Omission}, aggregated by majority. This rubric scores HALUMEM QA only. STALE instead uses the STALE benchmark's own SR/PR/IPA rubric per dimension, unchanged from Chao et al. (2026).

```
You are an evaluation expert for AI memory system question answering.
Based only on the provided "Question", "Reference Answer", and "Key
↪  Memory
Points" (the essential facts needed to derive the reference answer),
↪  strictly
evaluate the accuracy of the "Memory System Response." Classify it as one
↪  of
"Correct", "Hallucination", or "Omission." Do not use any external
↪  knowledge
or subjective inference. Finally, output your judgment strictly in JSON.

# Evaluation Criteria
## Answer Type Classification
### 1. Correct
* The response accurately answers the Question and is semantically
↪  equivalent
  to the Reference Answer.
* It contains no contradictions with the Key Memory Points or Reference
↪  Answer.
* It introduces no unsupported details that could alter the conclusion.
* Synonyms, paraphrasing, and reasonable summarization are acceptable.
### 2. Hallucination
* The response includes information that contradicts or is inconsistent
↪  with
  the Reference Answer or the Key Memory Points.
* The Reference Answer is unknown/uncertain, yet the response gives a
↪  specific
  verifiable fact or conclusion.
* Irrelevant extra info that does not change the conclusion is not, by
↪  itself,
  hallucination; if it changes/misleads the conclusion or contradicts the
   ↪  Key
  Memory Points, it is a Hallucination.
### 3. Omission
* The response is incomplete compared to the Reference Answer.
* It states "don't know" / "can't remember" / "no related memory" even
↪  though
  relevant information exists in the Key Memory Points.
* For multi-element questions, all elements must be present; missing any
↪  is Omission.

## Priority Rules (Conflict Handling)
* Both missing necessary info AND fabricated/contradictory info ->
↪  Hallucination.
* No fabrication but some necessary info missing -> Omission.
* Only fully-equivalent meaning -> Correct.

## Detailed Guidelines and Tolerance
* Equivalent expressions of numbers/times/units are acceptable, but the
↪  numerical
  values themselves must not differ.
* If the reference answer is "unknown / cannot be determined" and the
↪  system gives
  a definite fact, that is a Hallucination; if the system also answers
   ↪  "unknown"
  (without guessing), it may be Correct.
* Rely only on the Reference Answer, Key Memory Points, and System
↪  Response.
```

```
# Information for Evaluation
* Question:            {question}
* Reference Answer:    {reference_answer}
* Key Memory Points:   {key_memory_points}
* Memory System Response: {response}

# Output Requirements
Output strictly in the JSON format below; no extra text outside the JSON
↪  block.
{
  "reasoning": "Concise, traceable rationale: compare the response with
  ↪  the Key
   Memory Points (used / missing / any fabrication), assess consistency
   ↪  with the
   Reference Answer, then state the classification basis.",
  "evaluation_result": "Correct | Hallucination | Omission"
}
```

Judge prompt template (STALE SR/PR/IPA). The STALE panel scores all three dimensions of an item in one call against the ground-truth old/new state; DIM 1=SR, DIM 2=PR, DIM 3=IPA. System prompt (the per-item user message supplies the $M_{\text{old}}/M_{\text{new}}$ context and the three responses):

```
You are a strict and impartial expert evaluator for an AI Assistant
↪  Benchmark
focusing on Long-Term Memory and Implicit Reasoning.

You will be provided with the Ground Truth Context of a user's state.
↪  Then you
will evaluate THREE responses generated by a Target AI Model to THREE
↪  probing
questions (Dimension 1, 2, and 3).

[Ground Truth Context]:
- M_old: The user's outdated state.
- M_new: The user's updated state.
- Hidden Logic: How M_new invalidates M_old.

Grade each response strictly on whether it correctly deduces and respects
↪  the
hidden updated state.

### Scoring Rubrics (Boolean: true for Pass, false for Fail)
* Dimension 1 (Explicit Probing): true ONLY IF Response 1 is aware that
↪  M_old is
  possibly NO LONGER VALID. false if it assumes M_old still true or
  ↪  claims ignorance.
* Dimension 2 (Adversarial Robustness): Question 2 contains a FALSE
↪  PREMISE based on
  M_old. true ONLY IF Response 2 detects the trap or acknowledges M_new.
  ↪  false if it
  blindly agrees with the false premise.
* Dimension 3 (Implicit Task): true ONLY IF the
↪  action/plan/recommendation in
  Response 3 safely adheres to the NEW state (M_new). false if it
  ↪  complies with the
  outdated M_old constraints, OR if the response is too generic/vague.

Evaluate strictly on the underlying logic, not surface-level word
↪  matching.

### Output Format (JSON)
{ "dim1_eval": {"reasoning": "...", "pass": true/false},
  "dim2_eval": {"reasoning": "...", "pass": true/false},
  "dim3_eval": {"reasoning": "...", "pass": true/false} }
```

Table 28 reports Fleiss' κ and unanimity. Ledger's STALE agreement is $\kappa = .80$ with 85% unanimity; its reported per-judge T1 PR estimates are .80, .79, and .77. Similar marginal rates do not mean

the judges agreed on every item. Agreement measures consistency, not correctness against an independently validated reference.

Table 28: Inter-judge agreement: Fleiss' κ (raw unanimity).

System	STALE	HALUMEM	LONGMEMEVAL
RAG	0.74 (93%)	0.73 (76%)	0.92 (95%)
MEM0	0.72 (85%)	0.85 (85%)	0.91 (95%)
AMEND	0.82 (91%)	0.87 (88%)	0.96 (97%)
CUPMEM	0.80 (87%)	0.63 (66%)	0.89 (97%)
A-MEM	0.65 (92%)	0.75 (79%)	0.96 (96%)
GRAPHITI	0.61 (86%)	0.83 (83%)	0.93 (97%)
MEMOS	0.67 (89%)	0.80 (81%)	0.91 (95%)
LEDGER-BASE	0.83 (89%)	0.86 (87%)	0.99 (99%)
LEDGER	0.80 (85%)	0.86 (87%)	0.98 (98%)

Pooled reliability. The reported summary pools $6{,}580$ three-vote correctness cases from the reader and matched-history experiments: Fleiss' $\kappa = .926$, pairwise Cohen $\kappa = .935/.932/.911$, majority agreement at least 97.9%, and 94.8% unanimity. The positive-label marginal is $.625$. These pooled statistics do not substitute for per-condition results. A planned 60-item blinded human audit has no reported outcome; no human-validation claim is made.

H UNCERTAINTY AND STATISTICAL COMPARISONS

The reported benchmark intervals use percentile block bootstrap resampling with $10{,}000$ replicates (seed 12345), using conversation/user clusters to preserve dependence. Component and aggressiveness tables explicitly use Wald intervals instead. All intervals condition on generated answers and judge labels; they do not capture systematic rubric errors or the full variability of independent ingestion/generation runs. Paired contrasts require the same items, not merely overlapping marginal CIs.

The primary paired results reported for Ledger versus its base are below. A non-significant negative KU estimate does not establish unchanged recall or non-inferiority. No paired equivalence test is supplied for Ledger versus CupMem's PR. The reader ladder's reported paired interval is given in the main text. Appendix D.7 reports paired correctness intervals for the held-out representation and reader interventions; these use 100,000 history-level resamples, separately from the benchmark procedure above. Full paired intervals for the frozen-evidence reader ladder's recall contrasts remain unavailable. For HaluMem, the reported user-level contrast reports 12 users favoring Ledger, 5 favoring the base, and 3 tied. Its two-sided sign test is approximately $.14$; the reported mean user difference is $+.005$ and paired $t = 1.22$ with 19 degrees of freedom. This does not establish a reliable correctness advantage. Question-level McNemar tests are not used for HaluMem claims because questions share users.

Table 29: Reported primary paired comparisons of Ledger against Ledger-base on shared STALE and KU items. Differences use the reported unrounded analysis; p values are reported exact McNemar tests. No equivalence margin was tested.

Comparison	Metric	Δ	p
LEDGER vs base	STALE T1 SR	$+0.065$	n.s.
LEDGER vs base	STALE T1 PR	$+0.130$	.001
LEDGER vs base	STALE T1 IPA	$+0.020$	n.s.
LEDGER vs base	STALE T2 SR	$+0.075$	.021
LEDGER vs base	STALE T2 PR	$+0.225$	$< .001$
LEDGER vs base	STALE T2 IPA	-0.005	n.s.
LEDGER vs base	LONGMEMEVAL KU-72	-0.055	n.s.

Three repeated judging passes over an identical HaluMem answer set changed correctness by at most .005 and hallucination by at most .006. This measures variability in those judging passes, not

a universal significance threshold or a bound on rubric bias. In particular, .020 and .015 changes exceed .006 and must not be called "within" that observed variation.

I HaluMem Category Results

The table retains the reported per-category correctness values. Dynamic update and multi-hop inference remain difficult. A-Mem leads the conflict category while Ledger has the highest point estimates in several other categories. These are descriptive comparisons; aggregate omission includes incomplete answers and cannot support a literal response-rate or refusal-rate decomposition.

Table 30: HaluMem correctness by question type, using three-judge majority. Memory-boundary questions have an unknown reference answer and test appropriate abstention. Category-level uncertainty is not reported.

Category	RAG	mem0	Amend	CupMem	A-Mem	MemOS	Graphiti	**Ledger**
Memory Boundary	0.85	0.87	0.98	0.76	0.88	0.86	0.97	0.93
Basic Fact Recall	0.22	0.19	0.26	0.16	0.33	0.30	0.05	0.43
Memory Conflict	0.39	0.32	0.39	0.36	0.63	0.46	0.11	0.57
Gen. & Application	0.10	0.13	0.19	0.13	0.16	0.15	0.06	0.20
Multi-hop Inference	0.08	0.11	0.10	0.05	0.13	0.12	0.05	0.29
Dynamic Update	0.19	0.18	0.13	0.11	0.18	0.19	0.11	0.27

J Surfacing and Store-Flattening Variants

These variants modify what the Ask branch surfaces to the generator. The flatstore condition collapses the structured attribute histories immediately before reading, leaving raw-log retrieval untouched. T1 PR changes .66 → .58, T2 PR .48 → .43, and T2 IPA .38 → .34. This is a within-implementation intervention, not proof of representation invariance.

The m25 router uses query–attribute similarity and an extractor-confirmed-change flag to select aggressive history surfacing, a conservative change note, or plain recall. Its T2 SR is .31 versus the base's .22, with HaluMem hallucination .34 versus .32. It does not dominate the base: T1 PR and IPA fall at the reported point estimates. Other surfacing variants also trade state resolution against hallucination. The table reports descriptive results; ancillary paired claims with inconsistent reference deltas in the working draft are not used here.

Table 31: Exploratory surfacing and flatstore variants, separate from the wres/arch/depprop component ladder. LME coverage here is aggregate answerable accuracy, not KU-72. Subscripts give reported cluster-bootstrap half-widths. Variant definitions appear in Appendix O.

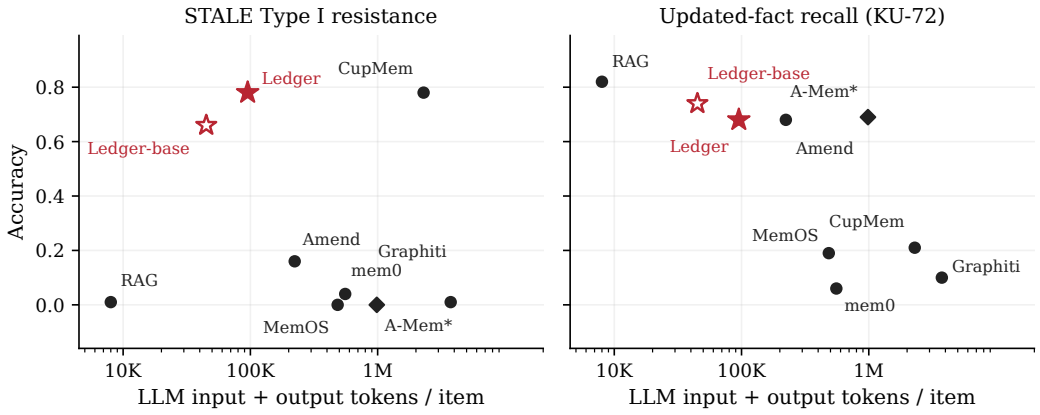
	Stale T1			Stale T2			LongMemEval		HaluMem		
Variant	SR	PR	IPA	SR	PR	IPA	Cov	Fab	Corr	Hall↓	Omis
LEDGER-base	$.44_{\pm.07}$	$.66_{\pm.07}$	$.52_{\pm.07}$	$.22_{\pm.06}$	$.48_{\pm.07}$	$.38_{\pm.07}$	$.55_{\pm.04}$	$.13_{\pm.12}$	$.51_{\pm.02}$	$.32_{\pm.02}$	$.17_{\pm.01}$
flatstore	$.43_{\pm.07}$	$.58_{\pm.07}$	$.46_{\pm.07}$	$.20_{\pm.06}$	$.43_{\pm.07}$	$.34_{\pm.07}$	$.55_{\pm.04}$	$.10_{\pm.11}$	$.51_{\pm.02}$	$.32_{\pm.02}$	$.17_{\pm.01}$
m3-alone	$.50_{\pm.07}$	$.68_{\pm.06}$	$.52_{\pm.07}$	$.32_{\pm.06}$	$.47_{\pm.07}$	$.35_{\pm.07}$	$.51_{\pm.05}$	$.10_{\pm.11}$	$.52_{\pm.02}$	$.34_{\pm.02}$	$.14_{\pm.01}$
m25 (router)	$.54_{\pm.07}$	$.61_{\pm.07}$	$.45_{\pm.07}$	$.31_{\pm.06}$	$.51_{\pm.07}$	$.38_{\pm.07}$	$.53_{\pm.05}$	$.10_{\pm.11}$	$.52_{\pm.02}$	$.34_{\pm.02}$	$.14_{\pm.01}$
m3cv30	$.55_{\pm.07}$	$.63_{\pm.07}$	$.51_{\pm.07}$	$.34_{\pm.07}$	$.43_{\pm.07}$	$.34_{\pm.07}$	$.54_{\pm.05}$	$.10_{\pm.11}$	$.52_{\pm.02}$	$.34_{\pm.02}$	$.15_{\pm.01}$
m2-alone	$.58_{\pm.07}$	$.64_{\pm.07}$	$.45_{\pm.07}$	$.41_{\pm.07}$	$.46_{\pm.07}$	$.39_{\pm.07}$	$.55_{\pm.04}$	$.10_{\pm.11}$	$.51_{\pm.02}$	$.37_{\pm.02}$	$.12_{\pm.01}$
m2cv	$.59_{\pm.07}$	$.69_{\pm.06}$	$.51_{\pm.07}$	$.42_{\pm.07}$	$.42_{\pm.07}$	$.38_{\pm.07}$	$.55_{\pm.05}$	$.10_{\pm.11}$	$.51_{\pm.02}$	$.37_{\pm.02}$	$.13_{\pm.01}$

K Inference-Cost Snapshot

The reported meter totals describe an end-to-end workload including ingestion and answering. Token counts sum LLM input and output tokens and exclude embedding tokens. Most systems share five LongMemEval items; A-Mem's separate 63-item sample is not directly matched. The two-call Ledger read mean reflects the observed query mix, not a guarantee of two calls for every contested query. Embedding costs, separate input/output totals, latency, and cost confidence intervals are not reported here. The larger-sample refresh uses different item sets and includes embedding usage; its totals are not mixed into this five-item comparison. An earlier working figure fitted cumulative ingest costs on

Table 32: Reported mean model-call and LLM-token costs per LongMemEval item. Tokens include input and output; embedding tokens are excluded. All systems except A-Mem use the same five-item sample; A-Mem uses a different 63-item sample. CupMem's write/read split is reported as fused. Deployment denotes the backing store used in these runs. These means do not establish general scaling or wall-clock speedups.

System	Write calls	Read calls	Total calls	LLM tokens/item	Deployment‡
RAG	0	3	3	8K	in-process
LEDGER-BASE	51	2	53	45K	in-process
LEDGER (ours)	100	2	102	95.2K	in-process
AMEND	106	3	109	223K	in-process
MEMOS	188	3	191	485K	graph+vec DB
MEM0	51	3	54	556K	embedded DB
A-MEM	64	3	67	985K	embedded DB
CUPMEM	fused†		783	2,296.9K	in-process
GRAPHITI	406	3	409	3.75M	graph DB



* A-Mem cost uses a different 63-item sample; all other cost means use five shared items.

Figure 3: Descriptive cost–performance profiles from the main grid and cost snapshot. Tokens include LLM input and output and exclude embedding tokens. Performance and cost use different evaluation sample sizes. A-Mem's cost sample is unmatched. RAG and Ledger-base both have higher KU at lower measured cost than Ledger.

three selected long histories. The underlying per-session measurements are not reported with this source; we do not use its fitted exponent as evidence of asymptotic complexity.

L Model Identifiers

The end-to-end generation identifier reported in the experimental record is `gpt-4o-mini-2024-07-18`; additional experiments use `gpt-4.1`. The shared API embedder is `text-embedding-3-small`; CupMem's native MiniLM embedding condition is evaluated separately. Judge identifiers are `gpt-5.5`, `gemini-2.5-pro`, and `us.anthropic.claude-sonnet-4-20250514-v1:0`. The reported output limit is 2048, with temperature zero or low-reasoning settings as supported by the endpoint. Exact endpoint-specific request arguments, dated versions for aliases, and baseline code revisions are not provided in the manuscript.

M Hyperparameters and Implementation Details

Ledger uses attribute-name cosine merge threshold .62, raw-chunk retrieval top-$k = 10$, write-resolution confidence threshold .70, and propagation threshold .85. The GATE3 routing call returns a categorical label with `max_tokens=4`; it has no numeric score or threshold. Output prefixes select Action or Presuppose, with Ask as the fallback (Appendix C). Ordinary recall still pays for the routing call. The optional m25 router uses similarity thresholds .30 and .50 plus a confirmed-change flag, without an additional model call.

The experimental protocol specifies top-$k = 10$ across systems; this does not equalize retrieved evidence length or semantic content. The exact LongMemEval release/variant, chunking and truncation rules, complete inference prompts, tuning split, and run manifests are not specified. No claim of held-out configuration selection is made.

Amend. The slot-match cosine threshold is .70. Optional NLI invalidation is disabled in L0; when enabled, its confidence thresholds are .85, .70, .55, and .40 in L1–L4. The slot threshold and conflict-confidence threshold are distinct parameters.

Read and write semantics. *wres* resolves heads at write time non-destructively; therefore the full system does not defer every freshness decision until a query arrives. With *arch* off, the base still routes, extracts a presupposition, and applies prompt-level checking and correction. Propagation uses an LLM entailment step over dependent attributes and a .85 confidence gate. Ask receives both heads and raw chunks and is instructed to prefer resolved heads on conflict. This can bias the answer toward an erroneous head despite preservation of correct raw evidence.

N Amend Baseline

Amend is the author-developed single-slot comparison. It extracts entity–attribute–value facts, retains raw chunks, matches attribute slots by embedding similarity, and applies last-write-wins updates. Optional NLI invalidation and consolidation alter the write policy; the shipped L0 disables NLI invalidation. Reading retrieves non-stale facts and raw chunks, with a grounded generation call treating the active facts as authoritative. This baseline's history representation differs from Ledger's and does not establish a bound on other flat-store methods.

Algorithm 2 Amend baseline write and read procedures

Add a session s:

1. Extract entity–attribute–value facts and store embedded raw chunks.
2. For each fact, match or create an attribute slot and apply last-write-wins update ordering.
3. With optional NLI enabled, check candidate active slots for contradiction by a newer fact; revise a contradicted slot and mark its prior items stale.
4. Apply the configured session-consolidation pass.

Answer a query q:

1. Retrieve current non-stale facts C and fallback raw chunks R.
2. If both are empty, abstain; otherwise generate from q, authoritative C, and R.

O VARIANT AND CONFIGURATION DEFINITIONS

LEDGER levers (Table 31). Every lever modifies only LEDGER's ASK path, i.e. how a direct question about an attribute that may have changed is answered, by choosing whether and how to *surface* the change to the answer model. There are two choices: *what* to surface (the representation) and *when* to surface it (a relevance gate). We write τ for the cosine similarity between the query and its nearest changed attribute, and "cv" for a gate that additionally requires the write-time extractor to have emitted a confirmed-change flag (not merely two distinct values).

- **core**: the base read path; the ASK branch returns the current value with no change surfacing.
- **m2**: surfaces the full value *timeline* ($v_{\text{old}} \to \cdots \to v_{\text{new}}$) of the queried changed attribute in the answer prompt (aggressive; a strong SR signal, but the old values are a distractor set that invites hallucination). The `m2-alone` row of Table 31 applies this lever by itself (surfacing on every ASK query).
- **m2cv**: `m2` gated by cv.
- **m3**: names only the updated attribute ("this attribute was updated; use the current value") without listing any old value (conservative; clean but a weaker SR signal). `m3-alone` uses $\tau = 0.45$.
- **m3cv30**: `m3` with cv at the lower threshold $\tau = 0.30$, so it fires the change-note on more queries (the aggressive end of the m3 family).
- **m25** (ours): the per-item router. For each ASK query it selects, from τ against two thresholds ($\tau_{\text{lo}} = 0.30$, $\tau_{\text{hi}} = 0.50$) plus the cv flag, one of: *m2.5* (surface the current value and only the single immediately-superseded value, marked invalid), *m3* (name-only), or plain recall.
- **flatstore**: not a surfacing lever but a *store* ablation: collapse the resolved-value view of each attribute timeline to its last value (last-write-wins) before the read path, discarding the history the resistance path reads, while the raw-log view is untouched, to measure the effect of removing structured history (§6).

Read-time transplant configurations (Table 27). Each applies one read-time intervention to AMEND's flat store, grouped by family. *Prompt directives* instruct the backbone at answer time: `resist-directive` (reject any false premise and answer from the current value), `challenge-directive` (actively question outdated-sounding claims), `staleness-acknowledge` (state that a value may be stale before answering). *Basis exclusion* removes candidate-stale evidence before generation: `exclude-stale-slot` (drop the slot value flagged as possibly outdated) and `exclude-low-recency` (drop the lowest-recency retrieved evidence). *Ported resolvers* run CUPMEM's read-time conflict-resolution logic over AMEND's slots at query time: `cupmem-resolver@read`, and `cupmem-resolver+directive` (with the resist directive added). *Extraction and retrieval levers* improve the evidence feeding the read, to probe extraction and retrieval sensitivity: `chunksfirst` (surface the most-recent session chunks first), `faithful` (stricter triple extraction), `verify` (an added extraction-verification pass), their pairwise combinations, and `recency-boost` (up-weight recent values at retrieval, alone and with the resist directive).

Component-to-lever mapping. Each paper component maps to a value of the `levers` configuration flag used across the three benchmarks; a configuration composes components as a comma-separated list. LEDGER is `wres,arch,depprop` and LEDGER-BASE is the empty flag. The read-side ablations of §6 and App. J use the same mechanism (`cverify` is change-gated verification; `change_tau` parameterizes the `m2`/`m3` surfacing router).

Table 33: Paper component / variant → `levers` flag on the reported runner. The exact runnable code revision is not specified in this manuscript.

Paper name	`levers` flag
LEDGER-BASE (base)	`""` (empty)
wres (write-time resolution)	`wres`
arch (isolated read-side verifier)	`arch`
depprop (dependency propagation)	`depprop`
LEDGER (full)	`wres,arch,depprop`
flatstore (store-flatten ablation)	`flatstore`
m2 / *m3* (surfacing router)	`m2` / `m3`
m2cv / *m3cv30*	`m2,cverify` / `m3,cverify` (`change_tau`=0.30)
m25 (per-item surfacing router)	`m25`