NOT ALL MEMORY UPDATES ARE EQUAL: A BENCHMARK FOR TYPED REVISION IN LONG-TERM LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory enables LLM agents to personalize assistance across interactions, yet evaluating memory updates requires more than checking whether the latest value can be recovered. We introduce RevisionBench, a diagnostic benchmark for the semantics of memory revision and use. It covers ten types of change, including factual correction, temporal replacement, preference evolution, temporary suspension, scope-limited overrides, goal lifecycle changes, and usage restrictions. Each target chain is paired with revision, candidate-discussion, and stable-confirmation histories, with queries probing current state, historical truth, and originally reported information. Our experiments show that systems can recover the correct current state while still failing to preserve the meaning of earlier evidence, and that competing candidates can be as challenging as actual revisions. Controlled studies further show that revision-aware interpretation improves state tracking, whereas explicit typed execution or graph storage does not consistently provide additional gains. RevisionBench provides a controlled testbed for diagnosing when memory systems conflate distinct revision semantics and for developing agents that model how, not merely whether, prior information is superseded.

1 INTRODUCTION

Long-term memory is a key component of large language model (LLM) assistants that provide coherent and personalized support across multiple sessions (Wu et al., 2025; Pan et al., 2025). By retaining user preferences, prior decisions, and relevant events, memory allows an assistant to interpret new requests in the context of earlier interactions rather than treating each request in isolation (Cheng et al., 2026; Zhang et al., 2026). This need has motivated methods for organizing and retrieving conversational memories, alongside benchmarks that evaluate their use in extended interactions (Pan et al., 2025; Hu et al., 2025). Yet retaining information is only part of the problem. As users correct earlier statements, revise plans, and introduce exceptions, an assistant must determine not only what it remembers, but also which information remains valid and under what conditions.

Not all updates have the same consequences for earlier information. Figure 1 shows that memory revision cannot be reduced to keeping the latest value. Factual correction and temporal replacement may lead to the same current state, yet differ in whether the earlier state remains historically valid. A capable memory system must therefore infer *how* new evidence changes the validity of old evidence.

Existing memory systems incorporate new information through dynamically linked notes (Xu et al., 2025), fact-level update operations (Chhikara et al., 2025), and temporal knowledge graphs (Rasmussen et al., 2025). Benchmarks likewise evaluate knowledge updates and temporal reasoning (Wu et al., 2025), evolving user preferences (Jiang et al., 2025), and selective forgetting (Hu et al., 2025). More recent work directly examines whether agents use current rather than superseded information (Patel, 2026; Fan et al., 2026). However, correctly recovering the current state does not establish whether a system distinguishes revisions with different implications for historical truth, scope, certainty, goal status, or permission. Our question is therefore more specific: *which semantic distinctions between revisions can an evaluation reveal, and which memory mechanisms help preserve them?* Addressing this question requires queries that distinguish these semantics, together with comparisons against histories containing competing candidates or no actual change.

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

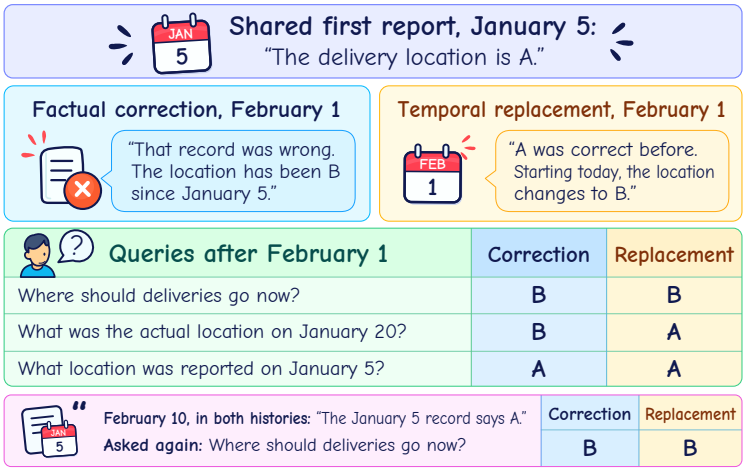


Figure 1: An illustrative contrast between factual correction and temporal replacement. Both produce the same current state, but imply different answers about the past. After the old record is quoted, the final query should still be answered with B in both histories.

To address this question, we introduce REVISIONBENCH, a controlled benchmark for evaluating both what a revision changes and what it leaves valid. REVISIONBENCH contains 300 target chains across 10 revision types. Each chain is instantiated under three matched history conditions and evaluated at three query positions, resulting in 2,700 primary QA instances. The three conditions correspond to an actual revision, discussion of alternative candidates without an operative change, and explicit confirmation of a stable state. Queries test the operative state and the start of its current uninterrupted interval. For correction and replacement, we additionally include 180 paired queries that distinguish the current state, historical truth, and the originally reported value. In total, REVISIONBENCH comprises 2,880 scored evaluation instances for each evaluated method. Evaluation across four LLM readers and six representative memory configurations shows that existing methods still struggle on REVISIONBENCH. Across all tested configurations, accuracy drops on genuine revisions compared with matched stable histories, with gaps ranging from 4.9 to 37.1 percentage points. This consistent degradation shows that revision handling is a major weakness of current memory systems.

Motivated by this limitation, we propose *Typed Revision Graph Memory* (TRGM), a revision-aware memory method designed to preserve and interpret changing information over time. TRGM represents conversational history as an atomic evidence graph rather than collapsing it into a single latest state. Evidence nodes retain the original statements and timestamps, while typed relations capture how later evidence changes the time, scope, or validity of earlier evidence. For example, a correction can revise historical truth without erasing the original report, whereas a replacement preserves the earlier value for the period in which it was valid. Scope relations additionally allow a more specific context to inherit from a broader applicable rule until a local override becomes active. At query time, TRGM reads the same graph according to the requested entity, reference time, scope, and answer type, allowing current state, historical truth, and original reports to be recovered from a unified memory representation.

Our main contributions are summarized as follows:

- **RevisionBench.** We introduce a controlled benchmark with 300 target chains, 10 revision types, and 2,880 scored evaluation instances for testing revision semantics across matched history conditions.
- **Typed Revision Graph Memory.** We propose TRGM, a revision aware memory method that preserves source evidence while explicitly interpreting how later statements change the validity of earlier information over time and scope.
- **Empirical findings.** Evaluation across four LLM readers reveals consistent weaknesses in existing memory systems, while TRGM achieves substantial gains on revision handling and semantic revision queries.

Table 1: Ten revision types in REVISIONBENCH. Each type is characterized by what new evidence changes and what information must remain recoverable. Examples are schematic.

Type	Short example	Semantic effect
Factual correction	*"The Jan. 5 location was B, not A."*	**Change:** supported past truth. **Keep:** the original report.
Temporal replacement	*"A was correct before. From Feb. 1, use B."*	**Change:** the state from the effective time. **Keep:** the earlier validity of A.
Preference evolution	*"I used to prefer coffee, but now I prefer tea."*	**Change:** the current preference. **Keep:** earlier preferences as history.
Scoped override	*"For work trips, use B; otherwise keep A."*	**Change:** the rule in the stated scope. **Keep:** the rule outside that scope.
Uncertainty change	*"I was considering A; I have now confirmed B."*	**Change:** certainty or commitment. **Keep:** the earlier tentative status.
Temporary suspension	*"Pause reminders until March 15."*	**Change:** activity status for a stated interval. **Keep:** the expiry and distinction from cancellation.
Goal cancelled	*"I am no longer planning the trip."*	**Change:** the goal becomes inactive. **Keep:** that no achievement was established.
Goal completed	*"I submitted the application."*	**Change:** the goal becomes completed. **Keep:** evidence of achievement and prior goal history.
Reactivation	*"Resume the reminders today."*	**Change:** the activity becomes active again. **Keep:** the preceding inactive interval.
Usage restriction	*"Remember my address, but do not use it for recommendations."*	**Change:** permission to use the information. **Keep:** the factual content itself.

2 REVISION SEMANTICS: WHAT CHANGES AND WHAT REMAINS

A query determines which part of the history matters. Let $H_{\le t} = \{(t_i, u_i) : t_i \le t\}$ denote the utterances available when a question is asked. We write a query as $q = (e, \tau, c, \pi)$, where e is the target entity, τ the reference time, c the scope, and π the requested answer, such as the current state, historical truth, original report, or usage permission. These fields are only conceptual; evaluated systems receive the conversation and natural language question. The gold answer is

$$a^*(H_{\le t}, q) = \mathrm{Read}_\pi\big(H_{\le t}; e, \tau, c\big). \tag{1}$$

The observation time t and reference time τ differ because new evidence can change what should be believed about an earlier time, reflecting the classical distinction between belief revision and world update (Katsuno & Mendelzon, 1992).

A revision changes some answers and preserves others. For a history H followed by a new utterance u, let $H \oplus u$ denote the updated history. For a query family $\mathcal{Q}$, define

$$\mathcal{C}_\mathcal{Q}(H, u) = \{q \in \mathcal{Q} : a^*(H \oplus u, q) \neq a^*(H, q)\}, \quad \mathcal{P}_\mathcal{Q}(H, u) = \mathcal{Q} \setminus \mathcal{C}_\mathcal{Q}(H, u). \tag{2}$$

These are the queries whose answers should change or remain unchanged, respectively.

For Figure 1, let q_{now}, q_{past}, and q_{report} ask for the current location, the actual location on January 20, and the January 5 report. Let $\mathcal{C}_{\mathrm{corr}}, \mathcal{P}_{\mathrm{corr}}$ denote the changed and preserved queries under correction, and define $\mathcal{C}_{\mathrm{repl}}, \mathcal{P}_{\mathrm{repl}}$ analogously. Then

$$q_{\mathrm{now}} \in \mathcal{C}_{\mathrm{corr}} \cap \mathcal{C}_{\mathrm{repl}}, \quad q_{\mathrm{past}} \in \mathcal{C}_{\mathrm{corr}} \cap \mathcal{P}_{\mathrm{repl}}, \quad q_{\mathrm{report}} \in \mathcal{P}_{\mathrm{corr}} \cap \mathcal{P}_{\mathrm{repl}}. \tag{3}$$

Thus, both revisions change the current state, only correction changes the historical truth, and neither changes the original report. Table 1 applies this change and preservation view to the ten revision types in REVISIONBENCH. The types describe the dominant revision effect, while candidate discussion and stable confirmation serve as matched non-revision controls.

Why current state accuracy is not enough. For a query family $\mathcal{Q}$, define two histories to be query equivalent when

$$H \sim_\mathcal{Q} H' \quad \Longleftrightarrow \quad a^*(H, q) = a^*(H', q) \quad \text{for all } q \in \mathcal{Q}. \tag{4}$$

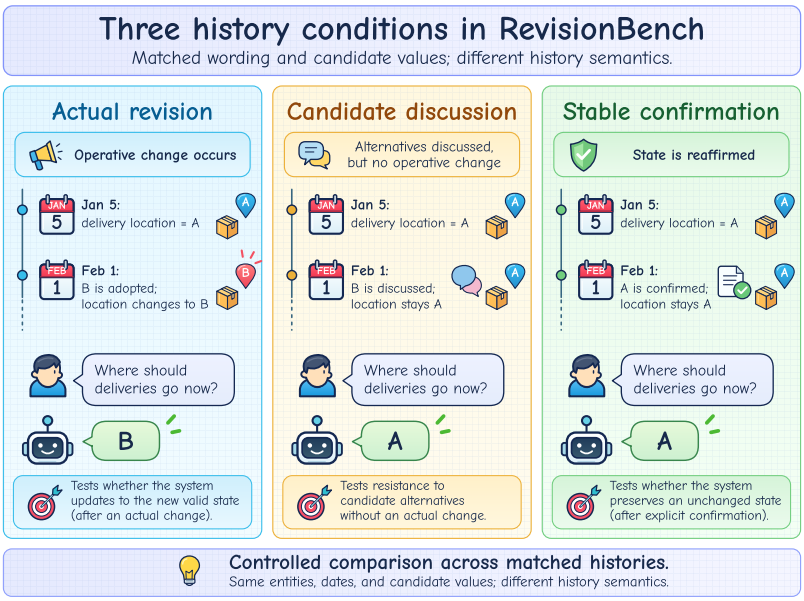


Figure 2: Matched history conditions in REVISIONBENCH. The histories keep entities, dates, and candidate values aligned, while the later evidence either changes the operative state, discusses an alternative without adopting it, or confirms the existing state.

The correction and replacement histories in Figure 1 are equivalent for the illustrated current state query, but they are not equivalent once the historical truth query is included. Let $\rho(H)$ denote the representation retained by a memory system. If $\rho(H) = \rho(H')$, but $H \not\sim_{\mathcal{Q}} H'$, then ρ is not sufficient for answering all queries in $\mathcal{Q}$. A decoder that receives only $\rho(H)$ and the query cannot recover a distinction that the representation has collapsed. Appendix A gives the corresponding elementary argument, including the randomized case.

3 REVISIONBENCH

3.1 CONSTRUCTION AND MATCHED CONTROLS

REVISIONBENCH contains 300 target chains and 10 revision types, embedded in multi-topic conversation histories. The chains form 60 history groups. Each chain is evaluated at three aligned query positions: after the first target event, after the final target event, and after a later mention of an obsolete value. We instantiate each chain under three matched history conditions, yielding 900 primary query instances per condition and 2,700 in total. The three conditions differ in whether the later evidence changes the operative state. In the **revision condition** (R), an operative change occurs. In the **candidate discussion control** (D), an alternative value is discussed, rejected, or mentioned without being adopted. In the **stable confirmation control** (S), the existing state is explicitly reaffirmed. Figure 2 illustrates the three conditions with matched entities, dates, and candidate values.

The two controls test different kinds of mistakes. Candidate discussion checks whether a system incorrectly treats a mentioned alternative as an actual change, while stable confirmation checks whether it can keep an unchanged state after new evidence is added. We use automatic checks for candidate values, timestamps, length, overlap, and aliases, together with manual review of a subset of the data. Appendix B gives further construction and validation details.

3.2 STATE, ONSET, AND PAIRED CONTRASTS

For the primary task, the answer projection π returns two quantities, $a^*(H_{\leq t}, q_j) = (s_j, d_j)$, where s_j is the complete operative state and d_j is the start date of its current uninterrupted interval. For example, if the current state is "suspended until March 15" and this suspension began on February 1, then $s_j =$ "suspended until March 15" and $d_j =$ February 1.

With N query instances, we report

$$A^s = \frac{1}{N}\sum_j \mathbf{1}[\hat{s}_j = s_j], \quad A^d = \frac{1}{N}\sum_j \mathbf{1}[\hat{d}_j = d_j], \quad A^{sd} = \frac{1}{N}\sum_j \mathbf{1}[\hat{s}_j = s_j \ \wedge \ \hat{d}_j = d_j]. \tag{5}$$

These measure state accuracy, onset accuracy, and joint accuracy, respectively. Answers are evaluated with a fixed strict scorer using predefined format and alias rules. For metric $m \in \{s, d, sd\}$ and history condition $g \in \{R, D, S\}$, let A_g^m denote the corresponding accuracy. We define

$$\Delta_S^m = A_S^m - A_R^m, \qquad \Delta_D^m = A_D^m - A_R^m. \tag{6}$$

Δ_S^m compares revision with stable confirmation, while Δ_D^m compares revision with candidate discussion. A positive gap means that the revision condition is harder. We report both the absolute accuracies and these paired gaps, since a small gap alone does not imply high accuracy.

3.3 Queries that separate revision semantics

To directly test the distinction between correction and replacement, we construct 30 paired cases. Each history is evaluated with three answer projections, $\pi \in \{\pi_{\text{current}}, \pi_{\text{history}}, \pi_{\text{report}}\}$, corresponding to the current value, historical truth, and the originally reported value. The 30 pairs contain one correction history and one replacement history, yielding 180 scored answers per method. These queries are designed so that correction and replacement can have the same current answer but different answers about the past, as illustrated in Figure 1. Additional typed-execution diagnostics use separate query contracts and are summarized in Appendix G. At test time, each method receives only the conversation available at the query cutoff and the natural language question. Revision type labels, history condition labels, gold answers, and scoring aliases are not provided.

4 Typed Revision Graph Memory

TRGM represents conversational memory as an atomic evidence graph whose typed revision relations capture how later information changes earlier evidence.

4.1 Atomic Revision Graph

TRGM represents an evolving conversation as a revision graph rather than a single latest-value memory. Given the observed history $H_{\le t}$, we construct $G_t = (V_t, E_t)$, where each node stores one atomic piece of evidence and edges record how pieces of evidence are related over time. An evidence node is written as $v_i = (e_i, a_i, x_i, t_i, c_i, r_i)$, where e_i is the entity, a_i the attribute being described, x_i its stated value, t_i the timestamp, c_i the applicable scope, and r_i points to the source record from which the evidence was extracted. The source record is retained so that later reasoning can distinguish what was originally stated from what is currently believed to be true.

The representation is atomic: one conversation statement may produce multiple evidence nodes when it contains different values or scopes. For example, "use A by default, but use B for work" creates separate nodes for the default and work-specific cases. Edges connect these nodes into an evolving history, $E_t = E_t^{\text{temp}} \cup E_t^{\text{source}} \cup E_t^{\text{scope}} \cup E_t^{\text{rev}}$, corresponding to temporal, source, scope, and revision relations. Temporal edges connect successive evidence about the same attribute, source edges preserve where the evidence came from, and scope edges connect more specific contexts to the broader contexts from which they can inherit a value. Typed revision edges record how later evidence changes earlier evidence. Because earlier nodes and their relations are retained, the same graph can support current-state, historical-truth, original-report, and scope-sensitive queries.

4.2 Typed Relations and Validity

Revision edges specify how new evidence changes the validity of earlier evidence. For two related nodes v_i and v_j with $t_i < t_j$, TRGM uses a frozen LLM to extract atomic evidence and revision relations from the observed conversation. For two related nodes v_i and v_j,

$$z_{ij} = \text{Rel}_M(H_{\le t}, v_i, v_j), \qquad (v_i, v_j, z_{ij}) \in E_t^{\text{rev}}, \tag{7}$$

where z_{ij} describes how the later evidence changes the validity of the earlier evidence.

To represent when and where evidence remains valid, each node v_i is associated with a validity region $\Omega_i \subseteq \mathcal{T} \times \mathcal{C}$, over time and scope. A typed revision edge updates these regions according to

$$\{\Omega_i\}_{i \le j} \leftarrow U_{z_{ij}}\big(\{\Omega_i\}_{i<j}, v_j\big). \tag{8}$$

Different relations therefore preserve different parts of the earlier history. For example, suppose a January 5 record says that the delivery location is A. If a February 1 statement says "starting today, the location changes to B," TRGM keeps A valid before February 1 and B thereafter. If it instead says "the January 5 record was wrong; the location has been B since January 5," TRGM revises the historical truth while retaining the original A record as evidence. Similarly, "use A by default, but use B for work" makes B applicable only to the work scope while leaving A valid elsewhere. Thus, revision edges encode not only that two pieces of evidence conflict, but also how the later evidence changes the time, scope, or status in which the earlier evidence remains valid.

Scope relations. Some scopes are related to broader scopes from which they can inherit a value. For example, suppose a rule says "use A for travel," and the benchmark contains a more specific scope called "work travel." Before any rule is stated for "work travel," that scope inherits A from the broader "travel" scope. If a later statement says "use B for work travel," then B takes precedence within the "work travel" scope. For a queried scope c, let $\mathcal{B}(c)$ denote c together with the broader scopes that are applicable to it. These relations are stored in E_t^{scope} and used during query readout.

4.3 Query-Specific Graph Readout

Given a query $q = (e, \tau, c, \pi)$, TRGM first collects evidence for entity e from the queried scope c and its applicable broader scopes $\mathcal{B}(c)$. The query projection π then determines how the graph is read. For current-state and historical-truth queries, TRGM resolves typed revision edges and scope relations to identify the evidence valid at reference time τ. For an original-report query, it instead follows the stored source evidence, so a later correction does not erase what was originally stated.

We write this query-dependent readout as

$$\hat{a}(q) = \mathrm{Read}_\pi\big(G_t; e, \tau, \mathcal{B}(c)\big), \tag{9}$$

where Read_π applies the revision and scope relations required by the requested projection. For the primary benchmark task, $(\hat{s}, \hat{d}) = \mathrm{Read}_\pi\big(G_t; e, \tau, \mathcal{B}(c)\big)$, where $\hat{s}$ is the predicted operative state and $\hat{d}$ is the start of its current uninterrupted interval. Thus, the same graph can return different answers to questions about the current state, an earlier truth, or the original report without discarding the underlying evidence.

4.4 Implementation

We implement TRGM with a frozen LLM and a deterministic graph executor. For each query cutoff, the LLM converts the available conversation into structured atomic evidence and predicts the relations needed to build the revision graph. The executor then applies the typed revision and scope rules to update evidence validity, after which the graph readout produces the requested answer. All extraction and relation prediction use fixed prompts and structured outputs shared across examples. Gold revision types and answers are used only for evaluation. Additional implementation and execution details are provided in Appendix C.

5 Experiments

5.1 Experimental Setup

Readers and data. We evaluate Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama3.1-8B-Instruct (Grattafiori et al., 2024) on all 300 target chains, with 900 primary queries in each of the revision (R), candidate discussion (D), and stable confirmation (S) conditions. GPT-4o-mini-2024-07-18 (OpenAI, 2024) and GPT-4.1-2025-04-14 (OpenAI, 2025) are evaluated on the same fixed 100-chain subset. We report state, onset, and joint accuracy, together with Δ_S and Δ_D.

Table 2: Full RevisionBench results on 300 target chains. S, D, and R denote stable confirmation, candidate discussion, and revision, respectively. State, Date, and Joint are accuracies (%); $\Delta_S = S - R$ and $\Delta_D = D - R$ are percentage-point gaps. BM25 and SeCom use $k = 5$. Mem0 denotes the timestamp-aware implementation used in our experiments. For Qwen with Graphiti, one missing stable-control history is counted as incorrect.

	State Accuracy			Paired Gap		Revision	
Method	S	D	R	Δ_S	Δ_D	Date	Joint
Qwen2.5-7B-Instruct							
Full Context	93.00	85.67	66.00	27.00	19.67	70.44	59.00
BM25	88.44	80.11	58.33	30.11	21.78	64.11	54.33
SeCom	89.67	81.33	60.67	29.00	20.67	67.78	57.67
A-Mem	91.11	84.56	61.33	29.78	23.22	71.00	59.11
SimpleMem	83.33	79.33	46.22	37.11	33.11	57.33	43.33
Mem0	85.56	72.89	61.67	23.89	11.22	72.89	57.78
Graphiti	84.67	67.22	38.00	46.67	29.22	50.00	37.11
TRGM	**97.78**	**87.89**	**91.11**	**6.67**	**-3.22**	**97.67**	**90.78**
Llama3.1-8B-Instruct							
Full Context	81.33	**78.44**	71.00	10.33	7.44	51.89	38.78
BM25	71.67	65.22	60.78	10.89	4.44	70.67	55.11
SeCom	73.44	67.56	64.22	9.22	3.33	75.11	59.44
A-Mem	75.78	68.56	66.00	9.78	2.56	74.89	56.44
SimpleMem	69.22	66.67	48.11	21.11	18.56	54.89	37.78
Mem0	43.00	29.67	30.67	12.33	-1.00	46.56	28.22
Graphiti	70.89	63.44	45.11	25.78	18.33	59.89	41.67
TRGM	**89.44**	78.00	**84.00**	**5.44**	**-6.00**	**96.78**	**83.11**

Baselines. We compare against Full Context, BM25 retrieval (Robertson & Zaragoza, 2009), SeCom (Pan et al., 2025), A-Mem (Xu et al., 2025), SimpleMem (Liu et al., 2026a), Mem0 (Chhikara et al., 2025), and Graphiti (Rasmussen et al., 2025). For Mem0, our implementation retains timestamps with stored memories; Graphiti follows its temporal knowledge-graph design. Additional TRGM variants and controlled ablations are introduced with the corresponding mechanism studies.

5.2 Overall Performance on RevisionBench

Table 2 shows that revision handling remains challenging even when the full history is available. Full Context avoids retrieval loss, yet reaches only 66.00% and 71.00% revision-state accuracy on Qwen and Llama. TRGM improves these scores to 91.11% and 84.00%, with joint state-and-onset accuracy of 90.78% and 83.11%. Additional results, including the 100-chain API-reader evaluation and full state, onset, and joint metrics, are reported in Appendix D.

Table 3 further shows that TRGM remains efficient at readout. It uses slightly fewer input tokens than BM25 and SeCom with comparable inference time, while Full Context processes roughly an order of magnitude more input. This suggests that TRGM can preserve revision-relevant evidence in a much more compact form. Complete call, token, and runtime statistics are reported in Appendix H.

Table 3: Readout cost over 1,500 queries.

Reader	Method	Input (M)	Time (s)
Qwen	Full Context	14.28	435
	BM25	1.28	141
	SeCom	1.31	120
	TRGM	1.19	136
Llama	Full Context	12.46	409
	BM25	1.15	125
	SeCom	1.18	103
	TRGM	1.12	106

The three history conditions also reveal clear differences between stable and revised histories. On Qwen, A-Mem drops from 91.11% on stable histories to 61.33% after revision, while Graphiti drops from 84.67% to 38.00%. TRGM reduces the stable-to-revision gap to 6.67 and 5.44 percentage points on Qwen and Llama. Its negative Δ_D further shows that candidate discussion can sometimes be harder than an actual revision, motivating both matched control conditions.

Table 4: Semantic accuracy (%) on the 30 correction–replacement pairs with GPT-4o-mini. Each cell contains $N = 30$ queries. Curr., Hist., and Report denote current value, historical truth, and original report.

	Correction			Replacement		
Method	Curr.	Hist.	Report	Curr.	Hist.	Report
Full Context	80.00	96.67	100.00	73.33	56.67	100.00
BM25	96.67	100.00	100.00	83.33	80.00	100.00
TRGM-Compact	100.00	100.00	100.00	100.00	73.33	100.00
TRGM-Typed	100.00	100.00	100.00	100.00	96.67	100.00

Table 5: Controlled studies of TRGM components. Results are state accuracy (%). For typed revision relations, the ablation replaces typed relations with a generic update rule. For scope relations, the ablation removes the applicable parent-scope edge. The two blocks use different diagnostic sets and should be compared only within each block.

Component	Reader	N	Full	Ablated	Gain
Typed revision	GPT-4o-mini	72	65.28	52.78	+12.50
	GPT-4.1	72	87.50	62.50	+25.00
Scope relation	GPT-4o-mini	12	83.33	0.00	+83.33
	GPT-4.1	12	100.00	0.00	+100.00

5.3 Performance across Revision Types

Aggregate accuracy hides substantial variation across revision types. On revised histories, TRGM reaches 98.89%, 100.00%, 100.00%, and 100.00% on factual correction, temporal replacement, preference evolution, and scoped override with Qwen, and 96.67%, 97.78%, 96.67%, and 86.67% with Llama. Errors are more concentrated in uncertainty changes, temporary suspensions, and goal cancellations; cancellation accuracy, for example, is 66.67% on Qwen and 60.00% on Llama.

The type breakdown also exposes weaknesses hidden by pooled scores. On Qwen, A-Mem reaches only 18.89% on temporary suspension and 30.00% on uncertainty change, compared with 82.22% and 81.11% for TRGM. These results show that different revisions place different demands on memory systems. Complete type-wise results are reported in Appendix E.

5.4 Revision Semantics beyond the Current State

Current-state accuracy alone does not reveal whether a system preserves the historical meaning of a revision. We therefore evaluate the paired correction and replacement queries from Section 3.3. Here, *TRGM-Compact* lets the reader interpret the retrieved revision evidence directly, whereas *TRGM-Typed* first applies explicit revision and scope relations to construct the state used for answering. Unless otherwise specified, TRGM refers to *TRGM-Typed* in our experiments.

Table 4 shows that methods can recover the current value and original report while still misinterpreting what was true in the past. This distinction is particularly clear for temporal replacement: historical-truth accuracy is 56.67% for Full Context, 80.00% for BM25, and 73.33% for TRGM-Compact, despite substantially higher accuracy on the corresponding current and report queries.

Explicit typed execution raises replacement historical-truth accuracy to 96.67% while retaining perfect current-value and original-report accuracy. This result directly supports modeling how later evidence changes the validity of earlier evidence, rather than representing a revision only by its latest value. Results for all four readers and additional controlled variants are reported in Appendix F.

5.5 What Enables Revision-Aware Memory?

We next examine two structures that are explicit in TRGM: typed revision relations and scope relations. Table 5 summarizes ablations of these two mechanisms.

Typed revision relations. We first compare full typed execution with a simplified executor that treats later assertive evidence as a generic update, without distinguishing whether it corrects, replaces, or changes the scope of earlier evidence. Typed execution improves state accuracy from 52.78% to 65.28% with GPT-4o-mini and from 62.50% to 87.50% with GPT-4.1. This shows that explicitly distinguishing how new evidence changes earlier evidence is useful when the query depends on revision semantics.

Scope relations. We next test whether scope relations are actually used during graph readout. For queries that require inheriting a value from a broader scope before a local override takes effect, removing the parent-scope relation reduces accuracy from 83.33% to 0% with GPT-4o-mini and from 100% to 0% with GPT-4.1. In contrast, direct queries about the later local value or the broader-scope value are unchanged by this removal. This selective drop shows that the scope relation contributes directly to the corresponding graph readout. Additional typed-revision and scope-relation results are provided in Appendix G.

6 Related Work

Benchmarks for long-term agent memory. Long-term memory benchmarks have evolved from evaluating recall and reasoning over very long conversations to broader assessments of temporal, episodic, personalized, and interactive memory capabilities. LoCoMo and LongMemEval established widely used testbeds for long-horizon conversational memory, including multi-session and temporal reasoning (Maharana et al., 2024; Wu et al., 2025). Subsequent benchmarks broadened the scope to episodic memory, incremental agent memory, dynamic user profiles, and multi-dimensional memory evaluation (Huet et al., 2025; Hu et al., 2025; Jiang et al., 2025; Tan et al., 2025). More recent work studies interactive and evolving memory through on-policy personalization, million-token conversations, latent user constraints, and state changes over time (Cheng et al., 2026; Tavakoli et al., 2026; Li et al., 2026b; Fan et al., 2026; Wu et al., 2026; Patel, 2026). RevisionBench complements these benchmarks by focusing on semantic distinctions between different kinds of memory revision.

Memory systems evaluated on long-term benchmarks. A parallel line of work develops memory mechanisms that improve performance on these benchmarks by changing how histories are compressed, structured, updated, and retrieved. Representative approaches use conversation segmentation and compression, self-evolving linked memories, fact-level consolidation, and temporal knowledge graphs (Pan et al., 2025; Xu et al., 2025; Chhikara et al., 2025; Rasmussen et al., 2025). Others introduce structured retain–recall–reflect pipelines, semantic compression, hierarchical memory management, or explicit temporal organization (Latimer et al., 2025; Liu et al., 2026a; Fang et al., 2026; Kang et al., 2025; Li et al., 2026a). Recent systems further explore adaptive memory management through self-evolving retrieval configurations (Liu et al., 2026b) and reinforcement-learned memory policies (Huo et al., 2026; Yan et al., 2026). These methods demonstrate that representation and retrieval choices strongly affect downstream memory accuracy. Our work is complementary, focusing on whether memory systems preserve the semantic distinctions required by revision.

7 Conclusion and Limitations

Conclusion. We introduced RevisionBench to study how long-term memory systems handle different kinds of information revision. Our results show that recovering the latest value is not enough: a system may answer the current state correctly while failing to preserve what was true earlier or what was originally reported. Across different memory designs, revision-aware interpretation improves state tracking, while explicit revision types alone do not guarantee better results. Overall, RevisionBench provides a way to evaluate whether memory systems preserve the meaning of change, rather than simply remembering the newest information.

Limitations. RevisionBench currently uses structured revision histories, and some semantic analyses cover only a subset of revision types. Future work can extend the benchmark to more natural conversations and broader revision scenarios.

AI USE STATEMENT

We used generative AI tools to assist with implementing methods, literature searches and manuscript editing. The authors reviewed the tested the experimental code and verified the references against their original sources. We take responsibility for the final manuscript, including its claims, references, code, figures, and experimental results.

REFERENCES

Jiayang Cheng, Dongyu Ru, Lin Qiu, Yiyang Li, Xuezhi Cao, Yangqiu Song, and Xunliang Cai. AMemGym: Interactive memory benchmarking for assistants in long-horizon conversations. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sfrVLzsmlf.

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.

Xinyi Fan, Miri Liu, Ruozhen Yang, Siru Ouyang, and Jiawei Han. Can agent memory systems track evolving state? *arXiv preprint arXiv:2608.19652*, 2026. URL https://arxiv.org/abs/2608.19652.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. LightMem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 98706–98729, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025. URL https://arxiv.org/abs/2507.05257.

Alexis Huet, Zied Houidi, and Dario Rossi. Episodic memories generation and evaluation benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 51409–51457, 2025.

Yupeng Huo, Yaxi Lu, Zhong Zhang, Haotian Chen, and Yankai Lin. AtomMem: Learnable dynamic agentic memory with atomic memory operation. *arXiv preprint arXiv:2601.08323*, 2026.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.14225.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors (ed.), *Belief Revision*, pp. 183–203. Cambridge University Press, 1992. doi: 10.1017/CBO9780511526664.007.

Chris Latimer, Nicolò Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*, 2025. URL https://arxiv.org/abs/2512.12818.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, et al. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21700–21720, 2026a.

Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. Locomo-plus: Beyond-factual cognitive memory evaluation framework for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25085–25100, 2026b.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents. *arXiv preprint arXiv:2601.02553*, 2026a. URL https://arxiv.org/abs/2601.02553.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Evolve-Mem: Self-evolving memory architecture via autoresearch for llm agents. *arXiv preprint arXiv:2605.13941*, 2026b.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Released July 18, 2024.

OpenAI. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 2025. Released April 14, 2025.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. SeCom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.05589.

Vedant Patel. Supersede: Diagnosing and training the memory-update gap in LLM agents. *arXiv preprint arXiv:2606.27472*, 2026. URL https://arxiv.org/abs/2606.27472.

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.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. In *International Conference on Learning Representations*, volume 2026, pp. 133343–133429, 2026.

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*, 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.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.12110.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yingyi Zhang, Junyi Li, Wenlin Zhang, Pengyue Jia, Xianneng Li, Yichao Wang, Derong Xu, Yi Wen, Huifeng Guo, Yong Liu, et al. Evoking user memory: Personalizing llm via recollection-familiarity adaptive retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 54178–54213, 2026.

A Query Separation and Representation Sufficiency

Section 2 views a memory representation through the queries it must support. The following elementary observation formalizes why a representation must preserve any distinction that changes a gold answer.

Observation 1 (A collapsed distinction cannot be recovered)**.** *Let ρ be a deterministic memory representation. Suppose two histories H and H' satisfy $\rho(H) = \rho(H')$ but have different gold answers to the same query, $a^*(H, q) \neq a^*(H', q)$. A deterministic decoder that receives only $(\rho(H), q)$ cannot answer both histories correctly. Under exact-answer scoring, a randomized decoder has average accuracy at most $1/2$ on a balanced pair containing one instance from each history.*

Proof. The decoder receives identical inputs for H and H'. A deterministic decoder must therefore return the same answer on both histories, which cannot equal two distinct gold answers. For a randomized decoder, let p be the common output distribution and let a and b be the two distinct gold answers. Its balanced-pair accuracy is $(p(a) + p(b))/2 \leq 1/2$. □

The correction and replacement example in Figure 1 provides a concrete instance. Both histories can share the same current value B, so a latest-value representation can make them indistinguishable. However, the historical-truth query separates them: a correction changes what should be believed about the earlier time, whereas a replacement preserves the earlier validity of A. The original-report query adds a second preservation requirement, because the source statement must remain recoverable even when later evidence changes historical truth. This is the representation-level motivation for retaining source evidence together with revision relations.

B RevisionBench Construction and Evaluation Details

B.1 Benchmark units and matched histories

RevisionBench contains 300 target chains distributed evenly across the ten revision types in Table 1, with 30 chains per type. The chains are organized into 60 history groups and embedded in multi-topic conversation histories. Each target chain is instantiated under the three matched history conditions used in the main text: revision (R), candidate discussion (D), and stable confirmation (S).

For each primary chain, questions are aligned to three query positions: (1) after the first target evidence, (2) after the final target event, and (3) after a later mention of an obsolete value. Across the three matched conditions, this yields 900 primary answers per condition and 2,700 primary answers per method. The 180 correction/replacement semantic questions are scored separately, for 2,880 evaluation instances when the two evaluations are counted together.

B.2 Matched-condition design

The three history conditions preserve the target entity, dates, candidate values, and surrounding conversational structure as closely as possible, while changing the semantic role of the later evidence.

Table 6: Evaluation units used in the reported experiments. Counts denote scored question-answer instances rather than independent conversations.

Evaluation	Size	Purpose
Primary local benchmark	300 chains; 900 answers per condition	State, onset, and joint accuracy under R, D, and S.
API reader study	Fixed 100-chain subset; 300 answers per condition	Replaces the answer model while keeping the memory evidence fixed.
Semantic paired study	30 correction/replacement pairs; 180 answers per method	Separates current value, historical truth, and original report.
Typed-relation diagnostic	72 revision queries per reader	Compares explicit typed execution with a generic update executor.
Scope-relation diagnostic	12 relation-dependent queries per reader	Tests inheritance from a broader scope before a local override.

Table 7: Roles of the three matched history conditions.

Condition	Later evidence	What the condition tests
R	An operative revision takes effect.	Whether the memory system changes exactly the information affected by the revision.
D	An alternative is discussed, rejected, or mentioned without adoption.	Whether a mentioned candidate is incorrectly promoted to the operative state.
S	The existing state is explicitly reaffirmed.	Whether additional evidence is handled without inventing a new state or onset.

The stable and candidate controls are complementary. Stable confirmation checks preservation under additional consistent evidence, whereas candidate discussion checks resistance to plausible but non-operative alternatives. This is why the main text reports both Δ_S and Δ_D rather than reducing the benchmark to a single control contrast.

B.3 Construction checks and scoring

Benchmark construction uses automatic checks for candidate values, timestamps, aliases, matched-history consistency, length, and lexical overlap. A subset of paired materials was additionally reviewed manually for semantic consistency and referential clarity. The strict scorer uses predefined answer formats and aliases and is shared across methods.

At inference time, every method receives only the conversation prefix available at the query cutoff and the natural-language question. Revision type labels, condition labels, gold answers, and scoring aliases are used only for evaluation. The conceptual query fields (e, τ, c, π) in Section 2 therefore describe the semantics of the task; they are not supplied as privileged structured labels to the evaluated systems.

When paired uncertainty intervals are used in controlled diagnostics, resampling is performed over the original history groups so that dependent query positions and matched conditions remain together. Reported percentages are computed from fixed denominators before display rounding.

C TRGM Implementation Details

TRGM implements Section 4 using a frozen language model for structured extraction and relation prediction, followed by deterministic execution over the resulting evidence graph. The graph retains evidence rather than replacing it in place, which allows the same memory to support current-state, historical-truth, original-report, and scope-sensitive readout.

C.1 GRAPH RECORDS AND RELATIONS

An evidence node has the form $v_i = (e_i, a_i, x_i, t_i, c_i, r_i)$, containing the entity, attribute, stated value, timestamp, scope, and source record. A single message can produce multiple evidence nodes when it contains multiple attributes or scopes. The implementation maintains the four edge families introduced in Section 4.1.

Table 8: TRGM graph components and their roles.

Component	Role in memory
Atomic evidence node	Stores one value-bearing piece of evidence together with time, scope, and source provenance.
Temporal relation	Orders successive evidence about the same entity and attribute.
Source relation	Keeps the original conversational record reachable after later revisions.
Scope relation	Connects a specific context to broader applicable contexts from which it can inherit a value.
Typed revision relation	Specifies how later evidence changes the time, scope, or status in which earlier evidence remains valid.
Query-specific readout	Resolves the graph for entity e, reference time τ, scope c, and projection π.

The validity representation $\Omega_i \subseteq \mathcal{T} \times \mathcal{C}$ allows different update relations to preserve different parts of the earlier history. Table 9 summarizes the semantic effects used by the executor. The table describes the effect of each relation family; source evidence itself remains stored in the graph.

Table 9: Semantic effects represented by typed revision relations.

Relation family	Effect on validity
Correction	Revises supported historical truth for the affected interval while retaining the original report as source evidence.
Replacement / preference evolution	Ends the earlier operative interval at the effective time and activates the new value afterward.
Scoped override	Activates the new value only in the stated child scope while preserving the broader value outside that scope.
Uncertainty change	Updates certainty or commitment from the effective point while retaining the earlier tentative evidence.
Suspension / reactivation	Changes activity over the relevant interval while retaining the preceding and subsequent lifecycle segments.
Goal completion / cancellation	Updates goal status while preserving the earlier goal history and the evidence that established the transition.
Usage restriction	Changes the permitted use of retained information without deleting its factual content.

C.2 EXECUTION SEQUENCE

For each query cutoff, the implementation performs the following steps.

1. Extract atomic evidence records from the available conversation.
2. Predict relations between relevant records, including typed revision and scope relations.
3. Apply the deterministic update operator in Equation 8 to obtain time- and scope-dependent validity regions.
4. For a query $q = (e, \tau, c, \pi)$, gather evidence for e from c and applicable broader scopes $\mathcal{B}(c)$.
5. Apply the projection-specific readout in Equation 9. Current-state and historical-truth queries resolve evidence valid at τ, whereas original-report queries follow the preserved source evidence.

Extraction and relation prediction use fixed prompts and structured output schemas shared across examples. Once evidence and relations are predicted, the graph executor and scoring pipeline are deterministic.

C.3 Compact and typed experimental variants

The experiments use two TRGM readout variants. *TRGM-Compact* presents the selected revision evidence directly to the reader and is the default TRGM configuration in the full 300-chain benchmark. *TRGM-Typed* first executes explicit revision and scope relations to materialize the state used for answering. Unless a table explicitly says TRGM-Typed, "TRGM" in the experimental results denotes TRGM-Compact. The semantic and mechanism studies use TRGM-Typed to isolate the effect of explicit relation execution.

D Detailed Three-Condition Results

This section expands the main benchmark table with state, onset, and joint accuracy for the three matched history conditions. In all tables below, S, D, and R denote stable confirmation, candidate discussion, and revision. The local studies contain $N = 900$ answers per condition.

Table 10: Qwen2.5-7B-Instruct: detailed local results (%).

	State			Onset			Joint		
Method	S	D	R	S	D	R	S	D	R
Full Context	93.00	85.67	66.00	94.89	46.00	70.44	88.56	45.00	59.00
BM25 ($k = 5$)	88.44	80.11	58.33	79.67	63.78	64.11	73.44	60.44	54.33
SeCom ($k = 5$)	89.67	81.33	60.67	86.00	62.89	67.78	79.00	60.89	57.67
A-Mem	91.11	84.56	61.33	94.22	73.56	71.00	86.00	69.67	59.11
SimpleMem	83.33	79.33	46.22	87.78	71.78	57.33	74.78	65.33	43.33
Mem0	85.56	72.89	61.67	85.11	41.44	72.89	75.78	38.56	57.78
Entity Policy	97.89	87.56	91.22	99.11	58.44	97.33	97.67	57.67	91.00
TRGM	97.78	87.89	91.11	99.11	59.56	97.67	97.56	59.00	90.78

Table 11: Llama3.1-8B-Instruct: detailed local results (%).

	State			Onset			Joint		
Method	S	D	R	S	D	R	S	D	R
Full Context	81.33	78.44	71.00	92.44	74.22	51.89	75.44	64.11	38.78
BM25 ($k = 5$)	71.67	65.22	60.78	85.78	78.89	70.67	64.11	59.11	55.11
SeCom ($k = 5$)	73.44	67.56	64.22	90.11	82.44	75.11	67.11	62.44	59.44
A-Mem	75.78	68.56	66.00	98.89	84.33	74.89	75.44	64.67	56.44
SimpleMem	69.22	66.67	48.11	89.78	73.11	54.89	64.67	55.67	37.78
Mem0	43.00	29.67	30.67	0.00	0.00	46.56	0.00	0.00	28.22
Graphiti	70.89	63.44	45.11	65.11	45.56	59.89	49.44	33.67	41.67
Entity Policy	89.44	78.67	84.56	99.33	69.44	96.78	89.11	62.89	83.78
TRGM	89.44	78.00	84.00	99.33	69.33	96.78	89.11	62.22	83.11

D.1 Fixed-evidence API reader study

The API study isolates the effect of the final answer model. The same 100-chain subset and fixed memory evidence are reused while the reader is replaced by GPT-4o-mini-2024-07-18 or GPT-4.1-2025-04-14. Each condition contains $N = 300$ answers.

Table 12: GPT-4o-mini fixed-evidence results (%).

	State			Onset			Joint		
Method	S	D	R	S	D	R	S	D	R
Full Context	94.33	83.00	89.33	39.33	33.00	92.00	37.00	31.67	85.67
BM25 ($k = 5$)	85.33	81.00	68.67	83.00	80.67	74.33	77.00	74.33	67.33
A-Mem	86.67	79.00	78.00	96.00	78.00	87.67	86.33	72.67	76.67
SimpleMem	84.00	81.00	56.00	83.33	66.67	65.00	74.00	63.00	50.33
Entity Policy	97.33	70.00	81.33	96.33	60.33	97.67	96.33	55.33	80.67
TRGM	97.33	70.00	81.33	96.33	61.33	97.33	96.33	56.67	80.33

Table 13: GPT-4.1 fixed-evidence results (%).

	State			Onset			Joint		
Method	S	D	R	S	D	R	S	D	R
Full Context	96.33	75.67	83.00	95.00	59.67	99.67	91.67	58.33	83.00
BM25 ($k = 5$)	91.33	72.33	72.00	85.67	68.00	83.33	83.33	64.00	71.67
A-Mem	95.33	73.67	75.33	98.67	71.00	96.00	95.00	68.00	75.00
SimpleMem	89.33	75.33	62.00	94.00	68.33	76.33	84.67	63.67	59.33
Entity Policy	95.33	81.33	87.33	98.00	80.00	98.67	95.33	76.67	86.67
TRGM	95.33	80.33	87.33	98.00	79.33	98.33	95.33	76.00	86.33

E Revision-Type Breakdown

The main text reports representative type-level results on the revision condition. Tables 14 and 15 give the complete ten-type revision-state breakdown for the principal external baselines and TRGM. Each type contains $N = 90$ answers in the local full benchmark. We abbreviate the types as follows: FC = factual correction, TR = temporal replacement, PE = preference evolution, SO = scoped override, UC = uncertainty change, TS = temporary suspension, GC = goal cancelled, GD = goal completed, RE = reactivation, and UR = usage restriction.

Table 14: Qwen2.5-7B revision-state accuracy (%) by revision type.

Method	FC	TR	PE	SO	UC	TS	GC	GD	RE	UR
Full Context	70.00	70.00	55.56	84.44	50.00	31.11	55.56	78.89	85.56	78.89
BM25 ($k = 5$)	71.11	74.44	58.89	75.56	55.56	13.33	34.44	64.44	56.67	78.89
SeCom ($k = 5$)	68.89	73.33	67.78	81.11	62.22	23.33	44.44	60.00	50.00	75.56
A-Mem	62.22	72.22	70.00	91.11	30.00	18.89	48.89	68.89	66.67	84.44
SimpleMem	45.56	60.00	60.00	67.78	36.67	6.67	32.22	55.56	48.89	48.89
Mem0	67.78	68.89	71.11	77.78	53.33	22.22	51.11	74.44	66.67	63.33
Graphiti	52.22	45.56	41.11	44.44	38.89	7.78	32.22	36.67	40.00	41.11
TRGM	**98.89**	**100.00**	**100.00**	**100.00**	**81.11**	**82.22**	**66.67**	**97.78**	**88.89**	**95.56**

The type profiles make the role of the taxonomy visible. Corrections, replacements, preferences, and scoped overrides are handled at high accuracy by TRGM on both readers, while uncertainty, suspension, and goal lifecycle changes remain more discriminative. The same ten type definitions are used for all methods; no type label is supplied at inference time.

Table 15: Llama3.1-8B revision-state accuracy (%) by revision type.

Method	FC	TR	PE	SO	UC	TS	GC	GD	RE	UR
Full Context	74.44	77.78	82.22	82.22	66.67	65.56	52.22	78.89	71.11	58.89
BM25 ($k = 5$)	78.89	72.22	75.56	80.00	56.67	45.56	40.00	56.67	46.67	55.56
SeCom ($k = 5$)	80.00	75.56	80.00	82.22	57.78	60.00	45.56	56.67	45.56	58.89
A-Mem	78.89	70.00	84.44	81.11	45.56	62.22	52.22	70.00	67.78	47.78
SimpleMem	52.22	45.56	54.44	55.56	34.44	44.44	48.89	63.33	42.22	40.00
Mem0	38.89	31.11	40.00	48.89	12.22	30.00	22.22	40.00	25.56	17.78
Graphiti	61.11	51.11	53.33	50.00	34.44	54.44	38.89	40.00	45.56	22.22
TRGM	**96.67**	**97.78**	**96.67**	**86.67**	**83.33**	**73.33**	**60.00**	**80.00**	**78.89**	**86.67**

F Complete Semantic Results

The paired semantic evaluation contains 30 correction/replacement pairs. Each method answers current-value, historical-truth, and original-report questions for both histories, yielding 180 scored answers per method and $N = 30$ in every type-by-projection cell. The controlled variants are included here because they help separate access to entity evidence from explicit revision interpretation: Entity History exposes chronological entity evidence, Entity Policy adds revision-aware interpretation, and Compact Plain uses the compact evidence representation without the full compact TRGM readout.

Table 16: Qwen2.5-7B semantic paired evaluation, strict accuracy (%).

	Correction			Replacement		
Method	Curr.	Hist.	Report	Curr.	Hist.	Report
Full Context	80.00	66.67	83.33	73.33	6.67	73.33
BM25 ($k = 5$)	76.67	90.00	83.33	76.67	56.67	83.33
Entity History	83.33	93.33	56.67	76.67	30.00	100.00
Entity Policy	100.00	100.00	100.00	100.00	90.00	100.00
Compact Plain	83.33	96.67	43.33	73.33	10.00	100.00
TRGM-Compact	100.00	100.00	100.00	100.00	50.00	100.00

Table 17: Llama3.1-8B semantic paired evaluation, strict accuracy (%).

	Correction			Replacement		
Method	Curr.	Hist.	Report	Curr.	Hist.	Report
Full Context	40.00	66.67	50.00	36.67	36.67	36.67
BM25 ($k = 5$)	60.00	40.00	60.00	66.67	23.33	40.00
Entity History	100.00	86.67	93.33	100.00	30.00	53.33
Entity Policy	100.00	60.00	86.67	100.00	23.33	70.00
Compact Plain	100.00	80.00	83.33	100.00	26.67	53.33
TRGM-Compact	100.00	46.67	66.67	96.67	23.33	46.67

Across readers, the three projections separate distinct requirements. A method can preserve the original report while still assigning the wrong historical validity to the reported value. The explicit typed executor is therefore evaluated on the projection where the correction/replacement distinction matters most rather than only on latest-value questions.

Table 18: GPT-4o-mini semantic paired evaluation, strict accuracy (%).

	Correction			Replacement		
Method	Curr.	Hist.	Report	Curr.	Hist.	Report
Full Context	80.00	96.67	100.00	73.33	56.67	100.00
BM25 ($k = 5$)	96.67	100.00	100.00	83.33	80.00	100.00
Entity History	100.00	100.00	100.00	100.00	36.67	100.00
Entity Policy	100.00	100.00	100.00	100.00	56.67	100.00
Compact Plain	100.00	100.00	100.00	100.00	60.00	100.00
TRGM-Compact	100.00	100.00	100.00	100.00	73.33	100.00
TRGM-Typed	**100.00**	**100.00**	**100.00**	**100.00**	**96.67**	**100.00**

Table 19: GPT-4.1 semantic paired evaluation, strict accuracy (%).

	Correction			Replacement		
Method	Curr.	Hist.	Report	Curr.	Hist.	Report
Full Context	100.00	100.00	100.00	100.00	96.67	100.00
BM25 ($k = 5$)	100.00	100.00	100.00	100.00	93.33	100.00
Entity History	100.00	100.00	100.00	100.00	100.00	100.00
Entity Policy	100.00	100.00	100.00	100.00	100.00	100.00
Compact Plain	100.00	100.00	100.00	100.00	100.00	100.00
TRGM-Compact	100.00	100.00	100.00	100.00	100.00	100.00
TRGM-Typed	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**

G Typed Revision and Scope Diagnostics

The additional diagnostics referenced in Sections 3.3 and 5.5 use fixed query sets designed to isolate relation execution. They are separate from the 300-chain primary benchmark and from the 30-pair semantic table above.

G.1 Typed revision relations

The typed-revision diagnostic holds the extracted evidence fixed and changes only how later assertive evidence is executed. The full executor distinguishes how a later statement changes earlier validity. The generic-update control uses a single update behavior rather than distinguishing correction, replacement, and scope-sensitive change. Each reader is evaluated on the same 72 revision queries.

Table 20: Typed-revision diagnostic. Values are state accuracy (%).

Reader	Typed execution	Generic update	Gain
GPT-4o-mini	65.28	52.78	+12.50
GPT-4.1	87.50	62.50	+25.00

The comparison keeps the underlying evidence set unchanged, so the gain measures the value of preserving distinctions among revision relations at execution time.

G.2 Scope inheritance and local override

The scope diagnostic uses 12 background groups with a parent scope and a more specific child scope. Three questions are asked after the same visible history: *child-before* asks for the child value before any child-specific override exists, *child-after* asks after the local override, and *parent-before* asks directly about the broader parent scope. The relation ablation removes the applicable parent-scope edge while leaving the source evidence unchanged.

The effect is selective: removing the parent relation changes the query that requires inheritance, while the direct child-after and parent-before queries remain unchanged. On the relation-dependent child-before queries, the full-minus-ablated difference is $+83.33$ percentage points for GPT-4o-mini with a group-bootstrap 95% interval of $[58.33, 100.00]$, and $+100.00$ points for GPT-4.1 with interval $[100.00, 100.00]$. The corresponding Holm-adjusted sign-flip p values are 0.00195 and 0.00098.

Table 21: Scope-relation diagnostic on revised histories. Each row contains $N = 12$ queries. Values are state accuracy (%).

Reader	Query	Full relation	Without parent	Difference
GPT-4o-mini	Child before local override	83.33	0.00	+83.33
	Child after local override	100.00	100.00	0.00
	Parent before local override	100.00	100.00	0.00
GPT-4.1	Child before local override	100.00	0.00	+100.00
	Child after local override	100.00	100.00	0.00
	Parent before local override	100.00	100.00	0.00

H Resource and Efficiency Accounting

The main text reports a compact readout-cost comparison. Table 22 provides the underlying call, token, and timing totals for the four methods whose recorded timing uses the same elapsed-inference convention. Each row covers the recorded 1,500-query readout run used in Table 3.

Table 22: Directly comparable readout accounting. Input and output are total LLM tokens; time is recorded elapsed inference time in seconds.

Reader	Method	Calls	Input tokens	Output tokens	Time (s)
Qwen2.5-7B	Full Context	1,500	14,280,786	41,985	435.0
	BM25 ($k = 5$)	1,500	1,280,973	42,020	140.8
	SeCom ($k = 5$)	1,500	1,310,285	42,381	119.6
	TRGM	1,500	1,186,348	46,866	136.1
Llama3.1-8B	Full Context	1,500	12,462,141	42,614	408.7
	BM25 ($k = 5$)	1,500	1,154,427	43,371	124.5
	SeCom ($k = 5$)	1,500	1,181,709	43,699	103.2
	TRGM	1,500	1,122,748	42,812	106.2

The readout totals show that TRGM uses a token budget comparable to the retrieval baselines and substantially less input than repeatedly reading the full conversation. The recorded elapsed inference times are likewise in the same range as BM25 and SeCom for both local readers.

For completeness, Table 23 reports LLM call and token totals from the recorded dynamic-memory runs. These totals are useful for accounting because dynamic systems may spend computation while constructing, updating, or consolidating memory in addition to final readout.

Table 23: Recorded LLM call and token totals for selected dynamic-memory runs. Mem0 denotes the timestamp-aware readout variant used in the main comparison.

Reader	Method	Calls	Input tokens	Output tokens
Qwen2.5-7B	A-Mem	46,200	66,532,011	16,456,274
	SimpleMem	14,449	25,495,661	7,389,216
	Mem0	1,500	2,642,481	41,704
	TRGM	1,500	1,186,348	46,866
Llama3.1-8B	A-Mem	46,200	61,502,478	15,120,518
	SimpleMem	15,627	20,906,058	8,329,509
	Mem0	1,500	1,875,525	43,644
	TRGM	1,500	1,122,748	42,812

Because construction and readout are recorded at different stages by different systems, Table 23 is presented as resource accounting rather than as a hardware-speed ranking. The matched elapsed-time comparison in Table 22 is the timing comparison used in the main text.

I Evaluation and Reproducibility Summary

For clarity, the reported experiments follow the same evaluation conventions throughout the paper:

- all methods are queried at the same conversation cutoff for a given benchmark instance;
- revision type labels, condition labels, gold answers, and scoring aliases are withheld from inference;
- the primary benchmark uses the same strict state/onset scorer for all methods;
- the semantic paired evaluation uses the same six cells per method: correction and replacement crossed with current value, historical truth, and original report;
- API reader studies use a fixed 100-chain subset and fixed memory evidence so that reader effects can be measured without changing the underlying memory contents;
- controlled mechanism studies keep the source evidence fixed while changing the indicated execution relation or graph edge.

These conventions align the main benchmark, semantic evaluation, and mechanism diagnostics around the same question: whether a memory system can recover the information that should change after a revision while preserving the information that should remain valid.