MemContract: Contract-Sensitive Evaluation for Mutable Agent Memory

Rui Li **Shuang Cao**
Hill Research

Abstract

A memory system can retrieve a fact accurately and still apply it after correction, deletion, or deprecation. MEMCONTRACT makes these state-management failures observable through six executable contracts and 1,080 multi-session tasks with distractor sessions, alias rotation, and controlled memory interventions. Across twelve systems, the ten structured memory systems span only 3.7 and 3.9 percentage points on store and retrieve, but 33.2 points on complete provenance; Zep/Graphiti achieves 69.4% balanced contract satisfaction, exceeding our controlled Graph Memory's 67.6%. Complementary interventions identify auditable mutation state as a practical design lever: withholding Graph's audit affordances lowers its score to 61.9%, while an append-only audit log and exposed status raise Hybrid RAG+KV from 62.4% to 66.3%, recovering most of its deficit. A symmetric backend-by-interface comparison measures the additional benefit of prompt and schema tuning under equal effort. Across ten structured systems evaluated on 240 downstream episodes, mutation scores retain a partial correlation of 0.83 with task completion after controlling for store/retrieve. On the correction-bearing episodes, superseded-policy use is 41.3% for RAG, 8.7% for Graph, and 8.0% for Zep/Graphiti. MemContract connects a failure to its prerequisite, state obligation, and tested interface repair, making mutable-memory evaluation actionable for agent design.

1 Introduction

An assistant handling a policy correction must do more than find the policy again. It must use the replacement for current decisions, retain an explicitly deprecated version for historical questions, honor deletion requests across retained representations, and identify the events that established the surviving value. Each transition creates a new correctness obligation beyond the earlier retrieval answer. For agents that carry information between sessions, the reliability of these transitions determines whether remembered information remains useful.

Long-term memory evaluations already test important temporal and conversational abilities. LONGMEMEVAL includes knowledge updates and temporal reasoning [Wu et al., 2025]; LOCOMO evaluates reasoning over extended conversations [Maharana et al., 2024]. Their answer-level scores capture whether a task succeeds. Diagnosing a failed update additionally requires observing whether the original write succeeded, whether the replacement became authoritative, whether obsolete content remained active, and whether the change was recorded. End-to-end agent benchmarks such as AgentBench, WebArena, and GAIA [Liu et al., 2024, Zhou et al., 2024, Mialon et al., 2024] combine these events with planning and tool use. An operation-level specification provides the complementary information needed to choose a repair.

MEMCONTRACT defines six observable operations: store, retrieve, revise, deprecate, erase, and prove-origin. Each has explicit state, prerequisites, and scored postconditions. The benchmark separates current-value readout from ordered provenance readout and distinguishes inactive historical

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

state from erased content. Two memories can return the same current value while retaining different revision histories or different recoverable copies of a deleted fact. MemContract distinguishes those states by evaluating the required transition and its subsequent observables. Its 1,080 tasks require persistence across distractor sessions and rotated aliases; prerequisite-conditioned scores locate whether failure begins before or during mutation. This specification makes the same diagnostic executable across retrieval, key–value, graph, and learned memory-management systems.

The decisive finding concerns access to mutation history and status. Among twelve systems evaluated under common budgets, Zep/Graphiti leads the balanced macro at 69.4%, and Memory-R1's 64.1% point estimate exceeds Mem0-graph's 63.7%. Across the ten structured systems, store/retrieve occupy a band narrower than four points, while provenance spans 33.2 points. Holding the storage layer fixed, withholding traversal, revision counters, and source annotations reduces Graph Memory by 5.7 points. Conversely, adding an audit log and exposing status increases Hybrid RAG+KV by 3.9 points. The resulting Graph–Hybrid gap is 1.3 points (95% paired CI $[-0.9, 3.5]$), compared with 5.2 under their original interfaces. A five-family slice excluding prove-origin retains this recovery: the log and status support revision and historical-state handling as well as provenance readout. Together with symmetric interface tuning, these interventions connect the score separation to a specific engineering action—exposing auditable mutation state.

These differences have a concrete consequence: on 138 downstream episodes containing corrections, RAG uses superseded policies in 41.3% of runs, versus 8.7% for Graph and 8.0% for Zep/Graphiti. Our contributions connect the specification to this consequence:

- A mutable-state evaluation object that distinguishes active values, inactive history, erased content, and ordered origin chains, with executable postconditions and prerequisite checks over 1,080 multi-session tasks.
- An intervention-based diagnosis of auditable mutation state: adding log/status access recovers most of Hybrid's deficit, withholding audit affordances impairs Graph, and a symmetric interface factorial measures tuning separately.
- Decision-relevant validation: mutation scores carry downstream information beyond recall, and correction episodes expose superseded-policy use. Contemporary-system profiles, six-backbone comparisons, workload reweighting, and human audits establish the evaluated scope.

2 Related Work

Memory architectures expose different ways to maintain persistent information. MemGPT manages hierarchical context [Packer et al., 2023]; Reflexion retains feedback [Shinn et al., 2023]; Generative Agents and MemoryBank organize experience for later use [Park et al., 2023, Zhong et al., 2024]. A-MEM links and evolves structured notes [Xu et al., 2025]. Mem0 extracts and consolidates facts, with a graph extension for relational information [Chhikara et al., 2025]; Zep's Graphiti represents temporally evolving relations [Rasmussen et al., 2025]. Memory-R1 learns memory-management operations and answer-time memory selection [Yan et al., 2026]. Ontology-guided conversational memory combines durable user facts, query enrichment, and hybrid retrieval [Cao and Li, 2026]. These mechanisms motivate concrete comparisons: preserving a fact, replacing it, retaining inactive history, and exposing its origin are separately testable properties even when implemented inside one memory manager.

LongMemEval tests information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. LoCoMo adds long conversational histories and event-oriented evaluation. MemContract makes state transitions themselves observable: a deprecated value must remain historically accessible while being excluded from active decisions; erasure is checked against a fixed residue-probe suite; provenance must contain the entire ordered creation/revision chain. Knowledge-update questions test whether the latest answer is recovered; the contracts additionally test whether inactive history is retained, deleted content remains recoverable, and the complete change history can be returned. Intermediate checks localize failed prerequisites, and paired changes to audit-state access identify a tested repair within a fixed backend. Long-context benchmarks [Bai et al., 2024, Shaham et al., 2022, Zhang et al., 2024] and multi-session conversation [Xu et al., 2022] supply complementary tests of information retention. Our external evaluations retain their question-

Algorithm 1 Observable memory contract

Require: Keyed state M, metadata d, ordered event log L, trusted timeline G

1: Store(k, v, d): require $k \notin M$; persist v as active, set $j = 0$, record creation.
2: Retrieve(q, S, h): read the current value in scope S; include inactive history only if h requests it. The optional single locator is ignored by the scorer. Leave state unchanged.
3: Revise(k, v_o, v_n): require active k and current value v_o; replace it with v_n, increment j, append the revision to L, and use v_n on subsequent active queries.
4: Deprecate(k, r): require active k; set inactive status and reason r; exclude it from active answers while retaining it for historical queries.
5: Erase(k): require a live or deprecated record; remove its value and value-bearing derivatives from the exposed boundary; return no value and pass the residue audit.
6: Prove-origin(k): for a retained record, return its ordered creation/revision chain of length $j + 1$, matching G in values, sessions, tool sources, and revision order.

answering objectives; the added value of contract measurement is localization and intervention guidance.

3 The MEMCONTRACT Benchmark

3.1 Operations and task construction

The interface vocabulary is create/read/update/soft-invalidate/hard-delete/audit-read. These map to the six contracts in Algorithm 1. Deprecation preserves content with inactive status; erasure removes content from the exposed memory boundary. Retrieve scores a current value, whereas prove-origin scores the events that established it. These operations share prerequisites: a failed initial write can invalidate several later operations.

Each family contains 180 tasks, with 60 each in easy, medium, and hard tiers. Episodes span 3–10 sessions (mean 5.4) across four domain templates. Every family includes intervening distractor sessions, alias/key rotation, and a final query whose answer is absent from the current session. Store tasks require recall after at least two unrelated sessions. Programmatic task construction is reviewed by three expert annotators ($\kappa = 0.82$). A two-parameter IRT pilot over twelve agent configurations calibrates difficulty; items with discrimination below 0.4 are regenerated. Difficulty means are -1.2, 0.1, and 1.6 logits, and mean discrimination is 1.1. Appendix I documents how delayed recall and alias rotation remove session-local shortcuts.

External coverage provides an empirical check on the vocabulary. Two coders, with third-coder adjudication, classified 412 memory-related failure reports from LangChain/LangGraph, LlamaIndex, AutoGen, and Mem0. Agreement was $\kappa = 0.79$; 376 reports (91.3%) map to one of the six operations. The residual concerns summarization, cross-user access control, and write conflicts (Appendix B). A separate 180-task audit finds 174/180 contract-fit agreement (96.7%) and 171/180 agreement on automatic pass/fail labels (95.0%); these measure different aspects of validity. Appendix I.1 gives the rubric and human-assisted evaluation.

3.2 State and scored boundary

Let $M : \mathcal{K} \rightharpoonup \mathcal{V} \times \mathcal{D} \times \mathcal{T}$ be a partial keyed state. Metadata d contains source session/tool, write time, status, revision index j, and deprecation reason. An absent key has no record; an active key supplies its current value; a deprecated key is readable only under an explicit historical query. A typed log records $\ell = (o, k, v_{\text{old}}, v_{\text{new}}, t, s, u)$ for operation, key, old/new values, time, session, and tool. Creation sets $j = 0$; each successful revision increments j and appends an event. The ground-truth event order breaks timestamp ties by operation order. Here $\mathcal{T}$ is the event-time domain; the record timestamp identifies its latest state transition, while write time in d identifies its creation.

In Retrieve(q, S, h), q is the query, S identifies the requested records, and h requests historical access; ordinary queries exclude deprecated records. Invalid mutation preconditions reject the operation without modifying state. Tasks execute operations in session order, and upstream failures remain failures in raw scoring. The state specification records creation at store time. The store/retrieve task scores check durable value correctness; history completeness is checked when a later operation

requires it. The retrieve scorer neither requires nor inspects a locator. Revise additionally checks the state update and corresponding log event, while prove-origin checks the complete chain returned to the caller. Thus excluding prove-origin removes chain readout, but retains mutation bookkeeping. Erasure applies to content-bearing audit entries, summaries, and scratchpad copies as well as the primary record. Non-content event metadata may remain; old values must not be recoverable through it. The log preserves event identity and order while erasure redacts value-bearing fields. The evaluator's separate ground-truth timeline is never agent-accessible.

The diagnostic object is the state trajectory, including its exposed history. Let $X = (M, L)$ and let $R_k(X)$ denote the current-value readout for key k. A complete revision from a 30-day policy to a 14-day policy and an overwrite that loses the creation event can have identical $R_k(X)$, yet only the complete history satisfies prove-origin. Deprecation adds a historical-read obligation; erasure adds a residue-removal obligation across retained representations. Algorithm 1 tests these distinct postconditions. The diagnostic sequence is therefore to check the prerequisite, identify the violated state obligation, and intervene on the interface capability that exposes that state.

Operation	State of a service-policy record	Subsequent observable
Store	Active: refunds within 30 days	Current query returns 30 days
Revise	Active: refunds within 14 days	Current query returns 14 days
Deprecate	Inactive: 14-day policy, reason recorded	Historical query only
Erase	No policy content at exposed boundary	No value; residue probes fail
Prove-origin	Creation followed by ordered revisions	Complete chain before erasure

Figure 1: One policy record, several distinct observables. The current value, historical status, recoverable residue, and ordered origin chain require separate checks. Prove-origin reads the retained history before erasure; it is not a step after erase.

3.3 Scoring and controls

A full-backend task scores one exactly when its operation-specific postconditions and later within-episode probes all pass; otherwise it scores zero. Blinded, stale, and delayed executions are separate controlled runs and never enter the full-backend success denominator. Blinding permits writes but returns empty reads; staleness supplies state from two sessions earlier; delay adds five seconds per operation within the same wall-clock cap. These conditions measure memory dependence, handling of obsolete reads, and sensitivity to access cost, respectively.

For erase, the fixed suite contains 25 probes: eight paraphrases, six multi-hop reconstructions, six contextual cues, and five compositional queries. Any successful content recovery fails the audit. The probe agent accesses only agent-visible interfaces and retained representations, while the trusted evaluator checks semantic recovery against the target. The resulting recovery rate measures exposed residue under this probe budget; it is not a deletion certificate. Algorithm 1 and Appendix F specify the boundary and its failure modes.

The primary summary is design-fixed balanced contract satisfaction, $B_s = \frac{1}{6}\sum_f \frac{1}{n_f}\sum_{i\in f}\bar{y}_{si}$, where $\bar{y}_{si}$ averages a task's three ordering seeds. For revise, we also report prerequisite satisfaction $q = P(Q)$, raw success $r = P(Y)$, and conditional success $c = P(Y \mid Q)$. Since $Y \Rightarrow Q$, $r = qc$. This exposes operation-specific failures without removing upstream failures from the primary score.

4 Systems and Evaluation Protocol

We evaluate twelve systems on the full GPT-4o-2024-05-13 pool. Seven controlled architectures span No-Memory (current-session context), a persistent 4,096-token Scratchpad, ChromaDB RAG [Lewis et al., 2020], Tombstone RAG, Editable KV, Hybrid RAG+KV, and Graph Memory. RAG uses ChromaDB 0.5.0 [Chroma, 2024], top-$k = 5$ cosine retrieval and `text-embedding-3-small`. Tombstone RAG adds timestamps, supersession flags, and recency decay ($\lambda = 0.18$/session). Editable KV exposes versioned JSON put/get/update/delete. Hybrid couples retrieval candidates to canonical KV values through union-then-verify resolution. Graph Memory uses Neo4j 5.19 with Cypher, source annotations, and revision counters [Robinson et al., 2015]. Five contemporary systems—A-MEM,

Mem0, Mem0-graph, Memory-R1, and Zep/Graphiti—use the same six-method adapter and scoring boundary. Their method-level interfaces are detailed in Appendix A.

The analytical cohort comprises ten structured persistent-memory systems, excluding No-Memory and the unstructured Scratchpad control. Scratchpad does persist; this exclusion distinguishes the analytical cohort from the full twelve-system evaluation. The principal runs share task instructions, planner backbone, temperature 0.0, and caps of 8,192 context tokens per turn, 256 KB serialized memory, 15 minutes, and $2 per episode. Backend-specific primitive calls implement the same operation requests. Adapter state and internal model calls consume the same memory, token, time, and cost budgets; an unsuccessful or budget-exhausted operation receives no success credit. The adapter translates requests and observes results without supplying missing history or repairing state on the system's behalf.

Three task-ordering seeds are retained together in paired task resampling (10,000 bootstrap replicates, 95% intervals); task IDs, not task–seed pairs, are the independent resampling units. Template-cluster intervals and a four-domain sensitivity analysis accompany key controlled-system contrasts. API inference is separated from local preprocessing and analysis: the itemized original evaluations consume 923.3 M prompt and 108.6 M completion tokens, with 138 A100-hours of local work on an eight-A100 80 GB node (CUDA 12.4, driver 550.54.15, PyTorch 2.3.1). Appendix C gives the components and defines the studies covered by this subtotal.

The seven controlled architectures are also evaluated on the same 360 task IDs with GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro 002, GPT-5.1, Claude Opus 4.5, and Gemini 3 Pro. Sampling without replacement fixes 60 tasks per family, with margins of 20 per tier and 15 per domain. These are marginal constraints rather than equal counts in every tier–domain cell. Each model receives the same semantic tool schema, budgets, and scoring, with vendor-specific call serialization normalized at the adapter boundary.

5 Results

5.1 Per-operation separation across twelve systems

Table 1 makes the contrast visible. Across the ten structured systems, store and retrieve span 3.7 and 3.9 points; prove-origin spans 33.2 points. Zep/Graphiti leads the macro with 69.4%, ahead of controlled Graph Memory at 67.6%, but its erase score is lower (50.7 versus 52.1). Memory-R1 exceeds Mem0-graph in revise and macro point estimates while trailing it in erase and provenance. These tradeoffs make the family profile useful for choosing an intervention. The original controlled top-three ordering applies to that seven-system comparison; contemporary systems change the ordering.

Table 1: Full-pool success (%) on 1,080 tasks with GPT-4o and three seeds. Original controlled systems show mean (seed SD); contemporary rows show means. Macro intervals are 95% paired task-bootstrap CIs. Each family contains 180 task IDs, with seeds clustered within task.

Method	Store	Retr.	Rev.	Depr.	Erase	Prov.	Avg. [95% CI]
No-Mem.	22.4(2.0)	19.8(2.1)	12.3(1.5)	8.7(1.2)	5.1(0.9)	3.2(0.7)	11.9 [10.3,13.5]
Scratch.	62.8(1.9)	58.4(2.2)	31.6(2.3)	22.4(1.8)	14.8(1.4)	11.3(1.1)	33.6 [31.5,35.7]
RAG	82.3(1.4)	85.7(1.2)	28.9(2.5)	19.6(1.7)	11.2(1.3)	15.7(1.6)	40.6 [38.9,42.3]
T-RAG	83.4(1.3)	85.2(1.2)	47.3(2.3)	38.4(1.9)	29.7(1.8)	22.6(2.0)	51.1 [49.0,53.2]
Edit. KV	83.1(1.3)	84.2(1.5)	62.7(2.1)	51.3(2.0)	43.8(1.9)	38.2(2.2)	60.6 [58.2,63.0]
Hybrid	83.9(1.2)	86.4(1.1)	64.8(1.9)	53.1(1.8)	45.7(1.8)	40.2(2.1)	62.4 [60.2,64.6]
Graph	85.6(1.1)	87.3(1.0)	71.4(1.8)	63.8(1.7)	52.1(2.0)	45.7(2.3)	67.6 [65.8,69.4]
A-MEM	84.2	86.1	55.4	42.6	33.9	32.6	55.8 [53.5,58.1]
Mem0	84.8	86.9	63.1	48.7	39.4	30.5	58.9 [56.7,61.1]
Mem0-graph	85.1	87.2	68.4	57.9	45.6	38.0	63.7 [61.5,65.9]
Memory-R1	85.4	87.6	72.1	60.3	44.8	34.4	64.1 [62.0,66.2]
Zep/Graphiti	86.0	88.1	74.3	68.2	50.7	48.9	69.4 [67.6,71.2]

Among controlled systems, Tombstone RAG improves on RAG by 18.4 points on revise and 18.5 on erase. Hybrid adds 2.1 and 1.9 points over Editable KV on these operations, respectively. Retrieval alone also misses a reversal: Scratchpad exceeds RAG on revise (31.6 versus 28.9) despite substantially worse retrieval. Graph's provenance score remains 45.7%, making complete history readout a demanding observable even for a system designed to expose it.

5.2 Conditioning separates prerequisite and mutation failures

The next question is where the separation arises in the execution. Table 2 conditions revise on successful storage and retrieval of the old value. Graph's conditional success is 74.6%, versus 69.3% for Hybrid and 34.1% for RAG. Conditioning narrows the Graph–Hybrid revise gap from 6.6 to 5.3 points, while substantial revision-specific failure remains. For example, under exhaustive prerequisite attribution, Graph's failed prerequisites account for approximately $(100 - 95.7)/(100 - 71.4) = 15.0\%$ of revise failures. This diagnostic share counts every failed prerequisite; it is distinct from a human assignment of a unique causal failure.

Table 2: Revise prerequisite satisfaction q, raw success r, and conditional success c (%). The final column is the algebraic prerequisite-failure share $(1 - q)/(1 - r)$, computed from rounded rates; it is not a separately estimated causal attribution.

System	q	r	c	Failure share
RAG	84.8	28.9	34.1	21.4
Tombstone RAG	89.2	47.3	53.0	20.5
Editable KV	92.9	62.7	67.5	19.0
Hybrid	93.5	64.8	69.3	18.5
Graph	95.7	71.4	74.6	15.0

The controlled reads provide a complementary diagnosis. Blinding measures reliance on persistent memory, whereas stale reads test handling of obsolete state; Appendix F reports reads per task, blinded drop per read, and stale success retention separately. Prerequisite conditioning localizes an invalid execution, while these interventions identify how the agent depends on the state it reads.

5.3 Exposing mutation state repairs the measured deficit

To test an actionable repair for history-dependent failures, we change exposed capabilities while holding each backend's storage layers, task IDs, prompts, and budgets fixed (Table 3). Withholding Graph's traversal, revision counters, and source annotations reduces its macro from 67.6 to 61.9. Adding an append-only audit log and exposing Hybrid's existing status field raises its macro from 62.4 to 66.3, recovering 3.9 points of the original 5.2-point Graph–Hybrid gap. The two directions test the same design hypothesis: history and status must be accessible to the agent performing the mutation. Hybrid gains that capability through a log over its existing retrieval/KV storage; Graph loses performance when access to its audit state is restricted.

Table 3: Capability intervention, full GPT-4o pool. Intervals are paired task-bootstrap 95% CIs. The residual Graph–audit-enabled-Hybrid contrast is 1.3 pp $[-0.9, 3.5]$, $p = 0.24$. The intervention recovers 3.9 of the original 5.2-point gap.

Exposed state	Six-family macro [CI]	Five-family macro
Graph: original	67.6 [65.8,69.4]	72.0
Graph: traversal/counters/sources withheld	61.9 [59.7,64.1]	—
Hybrid: original	62.4 [60.2,64.6]	66.8
Hybrid: audit log and status exposed	66.3 [64.1,68.5]	70.4

The five-family gap falls from 5.2 to 1.6 points after the Hybrid intervention. Removing prove-origin changes which outcomes are scored; adding the log and status changes which state the agent can use. Revision bookkeeping and historical-state handling remain required in the five-family score, so recovery on this slice connects the intervention to mutation behavior beyond chain readout. The practical repair is to expose the current status and ordered changes together with the value. The

1.3-point six-family residual measures that capability intervention; Section 6 separately tests how effectively existing capabilities are presented through prompts and tool schemas.

5.4 Downstream correction handling

We evaluate all twelve systems on 240 multi-session episodes: 60 retail/airline service, 90 web-navigation, and 90 scheduling/records episodes. Policy corrections, changed preferences or credentials, and mandated record updates make memory instrumental. The outcome is task completion under the final user requirements, not a MemContract label; planner, tools, GPT-4o backbone, and resource budgets are shared. Selected completion rates are 21.7% for No-Memory, 46.7% for RAG, 66.3% for Graph, and 66.7% for Zep/Graphiti.

Across the ten-system structured-memory cohort, balanced contract satisfaction correlates with downstream success at $r = 0.93$ (95% CI $[0.72, 0.98]$). The mutation-family mean, controlling for the store/retrieve mean, has partial $r = 0.83$ $[0.36, 0.96]$; the reverse partial correlation is 0.29 $[-0.44, 0.79]$ (Figure 2). Mutation performance carries additional information about task completion after adjusting for recall; the intervals describe uncertainty across the ten-system comparison. LongMemEval and LoCoMo correlations are 0.93 and 0.95, respectively. Contract measurement adds a decision beyond aggregate prediction: whether to repair a write/read prerequisite, the active-state transition, or historical-state handling. A-MEM performs better and Mem0 worse downstream than their relative contract macros suggest. Correction episodes supply a separate, directly interpretable readout of stale-policy use.

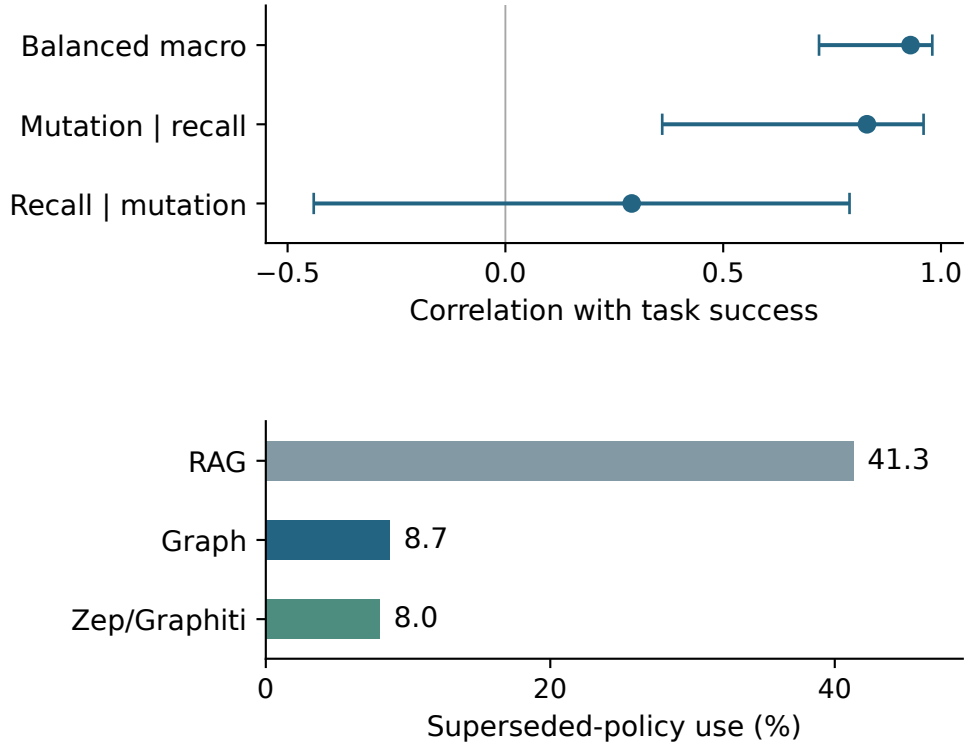


Figure 2: Downstream association across ten structured memory systems (upper panel, reported 95% intervals) and superseded-policy use on 138 correction-bearing episodes (lower panel). Mutation and recall denote their respective family means. The upper panel shows aggregate and adjusted estimates; the lower panel isolates the stale-policy outcome.

On those 138 episodes, superseded-policy use is 41.3% for RAG, 8.7% for Graph, and 8.0% for Zep/Graphiti. Among failed downstream episodes, 38.6% are attributed to contract-related errors:

stale values (17.2%), deprecated policies (9.4%), resurfaced deleted content (6.1%), and unattributable claims (5.9%). All four percentages use the failed-episode denominator. Appendix E separates these failure shares from correction-episode incidence and overall task completion.

6 Interface, Weighting, and Backbone Robustness

6.1 Symmetric backend-by-interface comparison

Four backends are evaluated under fixed interfaces, equal 40-person-hour tuning, and a maximal common schema without per-backend prompt search (Table 4). Every regime uses the same 1,080 tasks, three seeds, and budgets. The common schema exposes recency, status, version, and supersession fields wherever represented by the backend; it supplies no missing content or history. Tuning changes prompts and tool schemas within each backend family. The capability intervention in Section 5.3, in contrast, adds a log to Hybrid and withholds an affordance bundle from Graph.

Table 4: Symmetric interface factorial: macro success (%). Both systems in each comparison use the same regime. The tuned Graph–Hybrid gap is 4.1 pp [2.0,6.2], $p = 0.002$; the common-schema gap is 4.4 pp.

Backend	Fixed	Equal-effort tuning	Common schema
Tombstone RAG	51.1	54.2	52.7
Editable KV	60.6	63.5	62.1
Hybrid	62.4	65.0	63.8
Graph	67.6	69.1	68.2

Tuning benefits every backend. Relative to the fixed comparison, the symmetrically tuned Graph–Hybrid gap shrinks by 1.1 points, from 5.2 to 4.1. A task-level mixed-effects logistic analysis attributes 78.3% of explained fixed-effect variation to backend, 16.2% to interface regime, and 5.5% to their interaction (Appendix D). These proportions partition explained fixed-effect variation; the paired contrasts directly quantify the practical effects. Figure 3 places the capability and interface contrasts side by side.

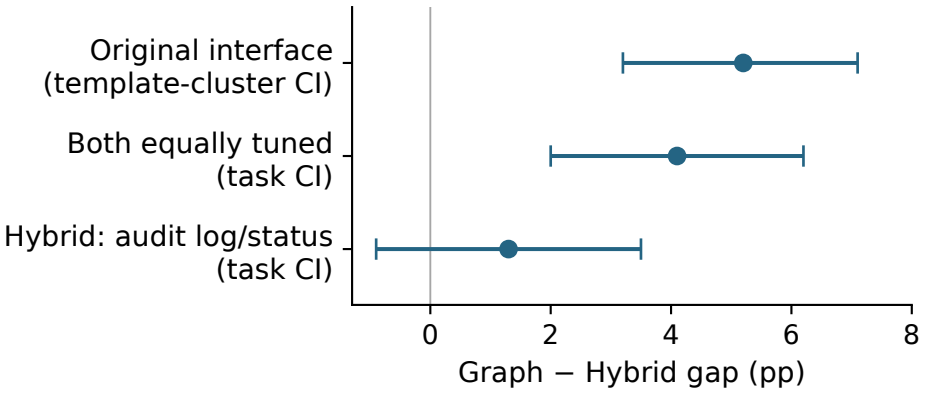


Figure 3: Graph–Hybrid paired macro contrasts. The original interface uses the more conservative template-cluster CI; the two interventions use their reported paired task CIs. Changing Hybrid's audit-state capability and tuning both interfaces test distinct mechanisms. Intervals are not equivalence tests.

6.2 Balanced and workload-weighted views

The balanced macro gives equal importance to all six contracts. Bifactor analysis assigns 63.8% common and 36.2% family-specific variation, with $\omega_h = 0.71$; general scores correlate with the

balanced macro at Spearman 0.98 across the original seven systems. The family profiles and conditional results preserve the information that this shared component does not capture.

Table 5 contrasts equal weighting, IRT-information weighting, and downstream operation-frequency weights. In family order, the latter are $(0.31, 0.38, 0.17, 0.07, 0.04, 0.03)$. They compress Graph–Hybrid from 5.2 to 3.2 points and Graph–RAG from 27.0 to 14.5 points. With the displayed controlled-system family means held fixed, Graph exceeds Hybrid and Hybrid exceeds Editable KV in every family. Consequently every strictly positive family-weight vector preserves that ordering; Dirichlet weight draws alone cannot estimate rank uncertainty. Sampling uncertainty instead enters the paired bootstrap (Appendix C).

Table 5: Sensitivity to the measurement target (%). IRT weights use item information; frequency weights emphasize store/retrieve. Weighting changes effect sizes while preserving the displayed controlled-system ordering.

Weighting	Graph	Hybrid	Editable KV	RAG
Equal family	67.6	62.4	60.6	40.6
IRT information	66.1	60.4	58.3	37.9
Operation frequency	79.8	76.6	74.9	65.3

6.3 Six backbones and external workloads

The controlled seven-system macro ordering is preserved on all six backbones (Table 6). Newer models improve absolute macros by 7–11 points and narrow the top-end gaps. Under GPT-5.1, structured systems span 2.8 points on store and 2.3 on retrieve, versus 32.6 on provenance. Across these backbones, higher absolute success coexists with substantial separation in history-dependent behavior.

Table 6: Paired macro gaps (pp) on the same 360 task IDs, seven architectures, three seeds per model. Complete family matrices appear in Appendices G and H.

Backbone	Graph–RAG	Graph–Hybrid
GPT-4o (2024-05-13)	27.1	4.6
Claude 3.5 Sonnet (2024-06-20)	26.6	5.4
Gemini 1.5 Pro 002	26.2	5.2
GPT-5.1	27.9	3.4
Claude Opus 4.5	27.6	3.5
Gemini 3 Pro	27.6	3.5

On the external 120-item LongMemEval selection and 100-item LoCoMo selection, the controlled Graph/Hybrid/RAG scores are 54.2/52.4/37.2 and 47.1/44.3/31.2, respectively. Graph–Hybrid narrows to 1.8 and 2.8 points. These comparisons change workload and scoring together, providing complementary evidence that effect sizes depend on the evaluated operations. Appendix J distinguishes the selected pools and analytical categories from the source benchmarks' official definitions.

7 Discussion

The six contracts describe English multi-session tasks in four domains, with summarization quality, access control, and concurrent writes outside the tested interface. The fixed 25-probe audit measures empirical residue recovery at the exposed boundary rather than certified deletion. Downstream associations concern ten structured systems, and the wide partial-correlation intervals warrant interpreting them alongside the concrete correction failures. Application-specific reliability assessment can combine the per-operation profile with workload frequencies and the exposed-state boundary.

8 Conclusion

MEMCONTRACT turns mutable-memory correctness into an inspectable sequence of state obligations. Exposing a log and status raises Hybrid from 62.4% to 66.3%, while restricting audit-state access reduces Graph from 67.6% to 61.9%. These complementary interventions identify accessible mutation history as a concrete design lever. Contemporary-system profiles and symmetric interface controls establish the relevance of this diagnosis across the evaluated designs. Downstream correction errors show why it matters to distinguish current values, retained history, erased content, and ordered origins. Checking prerequisites and state obligations turns a memory score into a concrete guide for improving the interface an agent uses.

Acknowledgments and Disclosure of Funding

This work received no external funding. The authors declare no competing interests.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://doi.org/10.18653/v1/2024.acl-long.172.

Shuang Cao and Rui Li. Ontology-guided long-term agent memory for conversational RAG. In *Proceedings of Machine Learning and Systems*, volume 8, pages 233–250, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/2fb4be70fc9668e9ec2c71b34fb127d4-Abstract-Conference.html.

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. doi: 10.48550/arXiv.2504.19413. URL https://arxiv.org/abs/2504.19413.

Chroma. Chroma: Version 0.5.0, 2024. URL https://github.com/chroma-core/chroma/releases/tag/0.5.0.

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*, volume 33, pages 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pages 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

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)*, pages 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, pages 9025–9049, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25ae35b5b1738d80f1f03a8713e405ec-Paper-Conference.pdf.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

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. doi: 10.48550/arXiv.2501.13956. URL https://arxiv.org/abs/2501.13956.

Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, 2 edition, 2015. URL https://www.oreilly.com/library/view/graph-databases-2nd/9781491930885/.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, 2022. doi: 10.18653/v1/2022.emnlp-main.823. URL https://doi.org/10.18653/v1/2022.emnlp-main.823.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

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*, pages 86809–86836, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, 2022. doi: 10.18653/v1/2022.acl-long.356. URL https://aclanthology.org/2022.acl-long.356/.

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*, volume 38, pages 17577–17604, 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 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)*, pages 12805–12825, 2026. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, 2024. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814.pdf.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, pages 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

A State, Adapters, and Independent Specification Check

The six-method adapter preserves each system's native storage and memory-management decisions. It maps an operation request to the exposed system interface and returns the resulting value, status, or history. Agent-visible summaries, caches, logs, and scratchpad content belong to the scored boundary and to the storage budget. Evaluator-only ground truth belongs outside that boundary. A deprecation that removes historical readability and an erase that merely hides a primary retrieval hit fail different postconditions. If a system lacks an operation, an attempted implementation through its native primitives is scored on its actual terminal state; the adapter does not create an oracle log.

Table 7: Memory implementations and the state exposed to the common contract. Method citations identify contemporary system designs; named database versions identify the controlled storage implementations.

System	Interface and relevant state
No-Memory / Scratchpad	Current-session conversation / persistent text read, append, overwrite; no structured provenance object supplied by the adapter.
RAG / Tombstone RAG	ChromaDB 0.5.0 similarity queries; Tombstone adds timestamps, supersession flags and recency reranking.
Editable KV / Hybrid	Versioned JSON records; Hybrid resolves retrieved candidates against a canonical KV record with recency and status.
Graph Memory	Neo4j 5.19 Cypher access to entities, relations, source annotations and revision counters.
A-MEM [Xu et al., 2025]	Linked notes and memory evolution; the adapter observes the note state produced by the memory manager.
Mem0 / Mem0-graph [Chhikara et al., 2025]	Fact extraction, consolidation and retrieval, with the graph variant retaining relational structure.
Memory-R1 [Yan et al., 2026]	Learned memory-management operations and answer-time selection; requested mutations are evaluated on the resulting memory.
Zep/Graphiti [Rasmussen et al., 2025]	Temporal entities and relationships, queried for current values and retained historical state.

In the capability intervention, Graph's backing database remains unchanged while traversal, revision counters, and source annotations are withheld from the agent interface. Hybrid retains its retrieval and KV layers while gaining an append-only event log and access to its existing deprecation-status field. The log uses the same typed events as Algorithm 1; erase redacts value-bearing content in all retained representations. Prompts and task IDs are unchanged. Consequently the within-system contrast measures the joint exposed capability bundle. Traversal, counters, and source annotations are withheld together, so the estimand is the bundle effect. Hybrid supplies complementary evidence using a log and status over retrieval/KV storage. Together, these interventions identify an implementable route to the history-dependent postconditions in Algorithm 1.

An independent Editable-KV implementation built from the state and operation specification reaches 59.9% on the full-pool protocol, compared with 60.6% for the original implementation. The independent score lies within the original macro interval [58.2,63.0], with a 0.7-point difference between point estimates. This specification-portability check shows that an independently implemented state interface yields a similar aggregate score. The original implementation's interval supplies context for the point-estimate comparison; the portability result concerns the independently realized contract interface.

B Task Population and External Failure Coverage

Table 8 summarizes the population and controls that define the evaluation. Equal family and tier allocations make the balanced macro a fixed measurement target; distractors and alias rotation require persistence beyond the current session.

The issue-report study codes the memory failure described in each report, rather than counting every downstream symptom as a separate operation. Reports come from the LangChain/LangGraph,

Table 8: Task population, construction, and evaluation units.

Field	Specification
Population	1,080 English multi-session tasks, six operation families
Family/tier allocation	180 per family; 60 easy, 60 medium, 60 hard
Sessions	3–10 per episode, mean 5.4
Domain structure	Four domain templates; 38 underlying templates
Cross-session controls	Distractors, alias rotation, answer absent from the final-session context
Annotation	Three expert task reviewers, $\kappa = 0.82$
IRT pilot	Twelve configurations; discard/regenerate discrimination below 0.4
Principal evaluation	Three ordering seeds per task, clustered by task for uncertainty
Residue probes	8 paraphrase, 6 multi-hop, 6 contextual, 5 compositional

LlamaIndex, AutoGen, and Mem0 issue trackers. Two coders assign one primary operation or a residual category, with disagreements adjudicated by a third coder ($\kappa = 0.79$). Store denotes failure to persist a required fact, retrieve failure to read the current fact, revise failure to replace it, deprecate failure to suppress inactive state while preserving history, erase recovery after deletion, and prove-origin failure to recover the creation/revision chain. Summarization distortion, cross-user access control, and write conflict remain separate residual categories. Table 9 shows that retrieval and revision account for most coded reports, while the residual categories identify distinct requirements outside the six contracts.

Table 9: Single-label coverage of 412 failure reports. Counts sum to 412; percentages are recomputed from integer counts and rounded independently. The six contract rows total 376/412 (91.3%).

Primary category	Reports	Percent
Retrieve	122	29.6
Revise	109	26.5
Store	51	12.4
Deprecate	46	11.2
Erase	28	6.8
Prove-origin	20	4.9
Summarization distortion	17	4.1
Cross-user access control	12	2.9
Write conflict	7	1.7

C Statistical Protocol and Estimands

For system s, task i, and seed a, let y_{sia} be the binary full-contract outcome. Each task's seed average is $\bar{y}_{si} = \frac{1}{3}\sum_a y_{sia}$. Family rates average $\bar{y}_{si}$ over the family's task IDs; the balanced macro averages the six family rates. A paired bootstrap resamples task IDs jointly across systems and retains all seed outcomes within each selected task. Template resampling instead draws whole templates, retaining their tasks, systems, and seeds. With only four domains, resampling domains is a sensitivity analysis for domain concentration, rather than a precise population-level uncertainty estimate.

For the original GPT-4o full-pool comparison, the template-cluster Graph–RAG gap is 27.0 pp [24.1,29.6] and Graph–Hybrid is 5.2 pp [3.2,7.1]. Domain sensitivity intervals are [21.9,31.8] and [2.4,7.8], respectively. Seven additional ordering seeds for RAG, Hybrid, and Graph give ten-seed macro means and SDs of 40.5±1.6, 62.2±1.5, and 67.4±1.3. These distinguish seed spread from uncertainty over the task population. The support of a three-run binary task average is exactly $\{0, 1/3, 2/3, 1\}$; no continuously supported per-task success distribution is assumed.

Conditional scoring uses the same task trials as raw scoring. Let Q denote that all required upstream write/read checks pass, and Y that the target operation succeeds. Then $Y \subseteq Q$ and

$P(Y) = P(Q)P(Y \mid Q)$. The exhaustive prerequisite-failure share among failed operations is $P(\neg Q)/P(\neg Y)$. This ratio measures where the execution was already invalid before the mutation. It is different from assigning one root cause after considering multiple failures in the same trace. The revise results in Table 2 use this algebraically defined diagnostic. From Table 1, the unconditioned, equal-weight means of the four displayed mutation-family rates are 58.25% for Graph and 50.95% for Hybrid, giving a 7.3-point raw difference. This displayed-rate arithmetic is distinct from prerequisite-conditioned scoring and from the single-family revise comparison. Table 10 gives the corresponding per-family diagnostics. The equal-family Graph–Hybrid conditional difference is 4.3 percentage points.

Table 10: Prerequisite-conditioned mutation-family success (%). All families use the same upstream write/read checks and raw full-contract outcomes as the primary evaluation. q is prerequisite satisfaction, r is raw success, and c is success conditional on passing the prerequisites; $r = qc/100$ in percentage units, up to displayed rounding.

Family	System	q	r	c
Revise	RAG	84.8	28.9	34.1
	Tombstone RAG	89.2	47.3	53.0
	Editable KV	92.9	62.7	67.5
	Hybrid	93.5	64.8	69.3
	Graph	95.7	71.4	74.6
Deprecate	RAG	77.5	19.6	25.3
	Tombstone RAG	78.4	38.4	49.0
	Editable KV	80.0	51.3	64.1
	Hybrid	78.7	53.1	67.5
	Graph	88.4	63.8	72.2
Erase	RAG	69.6	11.2	16.1
	Tombstone RAG	72.1	29.7	41.2
	Editable KV	76.4	43.8	57.3
	Hybrid	77.9	45.7	58.7
	Graph	84.3	52.1	61.8
Prove-origin	RAG	66.0	15.7	23.8
	Tombstone RAG	67.5	22.6	33.5
	Editable KV	75.9	38.2	50.3
	Hybrid	76.1	40.2	52.8
	Graph	80.3	45.7	56.9

Item-information weighting uses the calibrated two-parameter logistic model. For item i, $I_i(\theta) = a_i^2 p_i(\theta)(1 - p_i(\theta))$, and a weighted rate normalizes the selected information weights before summing outcomes. Operation-frequency weighting normalizes the six downstream event frequencies; an episode can contain several operation types, so these weights are not shares of disjoint episodes. Applying the reported frequency vector directly to the displayed family means yields Graph 79.769, Hybrid 76.608, Editable KV 74.905, and RAG 65.283%, which round to Table 5.

Weight-only rank stability follows directly from componentwise dominance. For $w_f > 0$ and $\sum_f w_f = 1$, if $p_{Af} > p_{Bf}$ for every family, then $\sum_f w_f(p_{Af} - p_{Bf}) > 0$. Thus Graph > Hybrid > Editable KV holds for every positive weight vector on the displayed controlled-system means. Rank uncertainty requires resampling outcomes as well as, optionally, weights; it cannot be inferred from Dirichlet weighting of fixed means.

The bifactor analysis distinguishes a shared success component from operation-specific components in the binary task-response matrix. Each item loads on a general factor and its assigned operation factor, rather than treating the six family scores as independent observations. The reported decomposition is 63.8% common and 36.2% family-specific variation, with hierarchical omega 0.71. No pair of operation families has a reported correlation above 0.93. The Spearman correlation of the general scores and balanced macros is 0.98 across the original seven systems. These descriptive results support retaining both balanced and family-specific views; the conditional table supplies the direct execution-level dependence diagnostic.

Table 11 removes each named family in turn. The Graph–Hybrid contrast remains about five points, so none of these families individually determines the controlled-system separation.

Table 11: Family withdrawal recomputed by equal weighting of the surviving displayed family rates. No outcomes or difficulty weights are changed. Values are approximate because the input rates are rounded; the five-family summary elsewhere uses the originally reported macro.

Removed family	Graph–RAG (pp)	Graph–Hybrid (pp)
Revise	24.00	5.04
Erase	24.32	5.08
Prove-origin	26.50	5.26

C.1 Compute accounting

The itemized API subtotal is 923.3 M prompt and 108.6 M completion tokens: main GPT-4o seven-system evaluation 414.2/48.1 M; Claude 3.5 subset replication 203.6/24.3 M; matched GPT-4o subset 74.8/9.1 M; Gemini 1.5 Pro replication 194.3/22.9 M; LoCoMo evaluation 36.4/4.2 M. This subtotal covers those five itemized studies; the contemporary-system, capability, factorial, downstream, and newer-backbone studies are separate experimental populations. Their evaluation sizes are specified in the corresponding sections, and all use the same per-episode cost and time caps.

Local processing totals 138 A100-hours: 46 for embedding generation over 4.2 M chunks, 31 for the residue-audit judge pipeline, 24 for statistical resampling, 19 for IRT and cluster analyses, and 18 for database warm-ups. Agent-facing model inference runs through vendor APIs; the local node is eight NVIDIA A100 80 GB GPUs with CUDA 12.4, driver 550.54.15, and PyTorch 2.3.1. These resources are separate from the per-episode API budget.

D Symmetric Interface Factorial

The factorial contains four backends, three regimes, 1,080 tasks, and three seeds: 38,880 task–seed executions before clustering. Fixed interfaces preserve the main comparison. Equal-effort tuning allocates 40 person-hours to each backend, including Graph. Common-schema evaluation exposes a shared vocabulary of recency, status, version, and supersession without per-backend prompt search. Backend-specific implementations remain responsible for keeping these fields correct. The study changes prompts, schema exposure, and retrieval resolution within a backend family; it does not add the audit-log capability tested in Section 5.3.

Table 4 reports all twelve cells. The appropriate tuned comparison is Graph 69.1 versus Hybrid 65.0, yielding 4.1 pp [2.0,6.2] and $p = 0.002$. Common-schema Graph 68.2 versus Hybrid 63.8 yields 4.4 pp. Relative to the fixed 5.2-point contrast, equal tuning recovers approximately 1.1 points. Comparing both systems within a regime holds interface-engineering effort fixed. The resulting 4.1-point tuned contrast and the 1.3-point capability-intervention contrast answer different design questions: improving use of existing state and adding access to mutation state, respectively.

A mixed-effects logistic model uses task-level binary success with backend, interface regime, and their interaction as fixed effects. The repeated outcomes are clustered by task and retain their template and seed dependence. The reported 78.3/16.2/5.5% decomposition refers to the backend, interface, and interaction portions of explained fixed-effect variation on the model scale. It is not a partition of all Bernoulli outcome variation, nor three independent causal effect sizes. Paired success-rate contrasts remain the primary quantities for engineering interpretation.

E Downstream Outcomes and Failure Denominators

The downstream pool consists of 240 multi-session tasks: 60 retail/airline service, 90 web-navigation, and 90 scheduling/records. The same planner, tools, model, and budgets are used for every memory system. A task succeeds when the final task-specific requirement is met; memory contracts are used for failure analysis, not as the success label. Corrections occur in 138 episodes. Completion rates, correction-error incidence, and shares of failed episodes therefore have different denominators.

Table 12: Downstream episode success across all twelve memory systems on the same 240-episode suite. Superseded-policy incidence uses the separate correction-bearing population described below.

System	Episode success (%)
No-Memory	21.7
Scratchpad	38.5
RAG	46.7
Tombstone RAG	52.1
Editable KV	57.6
Hybrid	61.0
Graph	66.3
A-MEM	60.4
Mem0	53.3
Mem0-graph	61.8
Memory-R1	62.9
Zep/Graphiti	66.7

Correlation analyses use ten system-level observations: RAG, Tombstone RAG, Editable KV, Hybrid, Graph, A-MEM, Mem0, Mem0-graph, Memory-R1, and Zep/Graphiti. No-Memory and Scratchpad are excluded by the structured-memory cohort definition, not by observed success. The recall covariate is the mean of store and retrieve; the mutation predictor is the mean of revise, deprecate, erase, and prove-origin. Partial correlation residualizes the predictor and downstream success with respect to the other family mean. The aggregate and partial intervals appear in Figure 2. A-MEM and Mem0 provide adjacent-order exceptions, demonstrating why the family profile is useful even when aggregate associations are strong.

The failure decomposition uses all failed downstream episodes as its denominator. Stale-value use contributes 17.2%, deprecated-policy use 9.4%, deleted-content resurfacing 6.1%, and unattributable claims 5.9%; together these account for 38.6%. The 41.3/8.7/8.0% correction rates instead measure superseded-policy incidence within correction-bearing episodes for each named system, including episodes that may also fail for another reason. These rates must not be summed with, or substituted for, shares of the failed-episode population.

F Residue Audit and Control Axes

The fixed audit issues 25 probes per erase task: eight paraphrases, six multi-hop reconstructions, six contextual cues, and five compositional queries. Majority judgment by three annotators determines whether a response recovers the erased content verbatim or as a semantic paraphrase. Recovery by any probe constitutes a failed audit. The union of successful attacks is therefore at least as large as the recovery rate of any one attack class on the same population. Audit pass is only one erase postcondition: correct state transition and no-result readout must also hold, so overall erase success can be much lower than one minus residue recovery.

Table 13: Fixed-suite audit on the original seven systems. Residue is recovery by any of 25 probes; detection is the reported catch rate for 140 injected contract violations; shortcuts counts flags among 12 planted opportunities. These are separate tests, not components of one success denominator.

System	Residue (%)	Violation detection (%)	Shortcuts caught
No-Memory	0.0	—	—
Scratchpad	18.7	67.2	7/12
RAG	27.4	71.8	9/12
Tombstone RAG	14.2	78.6	10/12
Editable KV	7.6	89.3	11/12
Hybrid	6.3	90.4	11/12
Graph	3.9	94.7	12/12

The evaluator, instrumentation, and ground-truth timeline are trusted. The probe agent can query exposed memory tools and any logs, summaries, or scratchpad records those tools return; it cannot inspect the evaluator's timeline, database storage outside the exposed boundary, or hidden model activations. Value-bearing log records are included in erasure, with only non-content event metadata retained. Table 14 relates each integrity property to its fault model and observable evidence.

Table 14: Contract evidence matrix. All rows assume a trusted timeline and scorer, and share the context, storage, cost, and time caps in Section 4. The cost column identifies the controlled resource boundary for each test.

Protected object	Fault/probe and enforcement boundary	Comparator and utility/cost control	Failure and evidence
Current value	Contradictory update; active-value read after distractors and alias rotation	Zep/Graphiti and Memory-R1; matched budgets and recall scores	Stale value; Tables 1, 2
Deprecated history	Obsolete policy; suppress active use while retaining historical access	Zep/Graphiti; Hybrid log/status intervention with storage fixed	Inactive policy reused; Tables 1, 3; Fig. 2
Erased content	Fixed 25-probe recovery; exposed values, logs, summaries and pad	Graph and Zep erase profiles; per-episode time/cost caps	Residue or invalid deletion; Table 13; Algorithm 1
Complete origin	Ordered creation/revision readout; exact fields and chain length	Zep/Graphiti and audit-enabled Hybrid; common schema	Missing or misordered event; Tables 1, 3
Contract transition	Injected pre/postcondition violations; evaluator checks trace	Five controlled persistent backends; same detector protocol	Undetected invalid transition; Table 13

The query-normalized results in Table 15 distinguish frequent memory access from the loss incurred when reads are blinded. Graph makes more reads per task than RAG but has a smaller drop per read and higher stale-state retention.

Table 15: Memory dependence and stale handling are separate axes. Reads are full-backend reads per task; drop/read is the full-minus-blinded percentage-point drop divided by that mean read count. Stale robustness is reported as a unitless success-retention ratio.

System	Reads/task	Drop/read (pp)	Stale robustness
RAG	3.2	7.1	0.65
Tombstone RAG	3.5	7.8	0.77
Editable KV	4.1	7.4	0.87
Hybrid	5.8	5.4	0.89
Graph	6.4	5.0	0.92

The absolute blinded drop measures reliance on persistent reads; dividing by full-run read count describes how that drop varies with the query pattern. It is not an effect estimate for deleting one additional query. For a system that makes no memory reads, this ratio is undefined and is not assigned zero. Stale robustness is the stale-run success rate divided by the full-backend success rate on the same task population; reads in the stale run return two-session-old values. The reported across-system correlation of the unnormalized dependence and stale-robustness axes is $r = 0.31$ $[-0.32, 0.75]$ across twelve systems; it is separate from the five-system normalized display. In the delayed condition every memory operation incurs five seconds within the 15-minute episode cap, making excess access a direct budget cost.

G Cross-Backbone Subset and Family Results

The matched subset contains 360 tasks sampled without replacement using a fixed seed, with 60 per family. Within each family, tier margins are 20 easy/20 medium/20 hard and domain margins are 15 per domain. Subset and full-pool IRT difficulty means are 0.163 and 0.167 (KS $D = 0.021$, $p = 0.99$); mean discrimination is 1.10 in both. The reported failure-type comparison is $\chi^2(4) = 2.31$, $p = 0.68$. Because the subset is part of the full pool, these are distribution-agreement diagnostics. Matched GPT-4o reruns, rather than comparisons against different task populations, anchor the cross-backbone gaps.

Claude uses the 2024-06-20 Claude 3.5 Sonnet version and GPT-4o the 2024-05-13 version. Gemini uses `gemini-1.5-pro-002`. The newer-model comparisons use GPT-5.1, Claude Opus 4.5, and Gemini 3 Pro, with the same semantic tool interface and temperature 0.0. Table 6 enumerates all six backbones and their reported paired gaps. Tables 17 and 18 resolve the Claude and matched GPT-4o comparisons by family: similar store/retrieve rates coexist with larger differences on revision and historical-state operations. Table 16 gives the complete newer-backbone family means.

Table 16: Per-family success (%) for the three newer backbones on the matched 360 task IDs, with three ordering seeds and the same tool interface, prompts, scoring, and resource caps. Avg is the equal-weight mean of the six families.

Method	Store	Retr.	Rev.	Depr.	Erase	Prov.	Avg
GPT-5.1							
No-Mem.	32.1	29.0	23.2	17.1	11.6	9.4	20.4
Scratch.	72.5	68.6	42.9	32.4	22.8	15.8	42.5
RAG	91.0	93.0	40.8	29.6	17.6	21.4	48.9
T-RAG	91.5	93.2	58.9	49.7	38.4	22.9	59.1
Edit. KV	92.0	93.7	75.2	64.8	54.4	37.5	69.6
Hybrid	92.8	94.3	78.4	69.0	55.3	50.6	73.4
Graph	93.8	95.3	82.1	74.6	61.0	54.0	76.8
Claude Opus 4.5							
No-Mem.	32.8	29.5	23.9	18.4	11.9	9.5	21.0
Scratch.	73.1	69.2	43.5	32.9	22.5	16.8	43.0
RAG	91.1	93.4	41.3	30.1	18.5	22.6	49.5
T-RAG	92.0	93.6	59.2	50.3	38.8	23.1	59.5
Edit. KV	92.6	94.1	75.6	65.3	54.8	38.2	70.1
Hybrid	93.1	94.5	78.9	69.3	55.5	50.3	73.6
Graph	93.8	95.6	82.2	74.8	61.2	55.0	77.1
Gemini 3 Pro							
No-Mem.	31.7	29.4	23.0	17.6	11.8	9.5	20.5
Scratch.	72.6	69.1	43.2	32.6	22.7	16.6	42.8
RAG	90.8	93.4	41.4	30.2	18.2	21.8	49.3
T-RAG	91.4	93.1	58.8	49.9	38.2	22.6	59.0
Edit. KV	91.9	93.5	75.0	64.9	54.5	37.8	69.6
Hybrid	92.6	94.2	78.6	68.8	55.7	50.5	73.4
Graph	93.6	95.1	82.4	74.9	61.3	54.1	76.9

H Gemini 1.5 Pro Family Results

Table 19 uses the same 360 IDs and three ordering seeds. The Graph–RAG gap is 26.2 pp [23.7,28.7], and Graph–Hybrid is 5.2 pp [3.4,7.0]. The corresponding GPT-4o Graph–RAG interval is [25.3,28.9], and Claude's is [24.1,29.2]. These are between-backend contrasts within each model. They do not test a difference between model backbones.

Table 17: Per-family success (%) on the matched 360-task subset with Claude 3.5 Sonnet. Cells are means (seed SD) over three ordering seeds.

Method	Store	Retr.	Rev.	Depr.	Erase	Prov.	Avg
No-Mem.	21.4(2.1)	18.9(2.2)	11.1(1.6)	8.1(1.3)	4.8(1.0)	2.9(0.8)	11.2(1.5)
Scratch.	60.6(2.0)	56.3(2.3)	29.4(2.4)	20.7(1.9)	13.5(1.5)	9.8(1.2)	31.7(1.9)
RAG	80.8(1.6)	83.4(1.4)	26.1(2.7)	17.3(1.9)	9.8(1.5)	13.9(1.8)	38.6(1.8)
T-RAG	81.6(1.4)	83.0(1.3)	44.1(2.4)	35.7(2.0)	26.8(1.9)	20.4(2.1)	48.6(1.8)
Edit. KV	81.2(1.5)	82.0(1.6)	59.8(2.3)	48.9(2.1)	41.2(2.0)	35.4(2.4)	58.1(1.9)
Hybrid	82.1(1.3)	84.3(1.2)	61.9(2.0)	50.2(1.9)	43.1(1.9)	37.4(2.2)	59.8(1.8)
Graph	83.9(1.3)	85.1(1.2)	68.7(2.0)	61.2(1.9)	49.3(2.2)	43.1(2.5)	65.2(1.8)

Table 18: Matched GPT-4o family results on the same 360 task IDs. Cells are means (seed SD); these results anchor the paired backbone comparison.

Method	Store	Retr.	Rev.	Depr.	Erase	Prov.	Avg
No-Mem.	22.1(2.0)	19.5(2.1)	12.4(1.6)	8.3(1.3)	5.0(0.9)	3.4(0.8)	11.8(1.5)
Scratch.	62.3(1.9)	58.0(2.2)	32.0(2.3)	22.8(1.8)	14.6(1.4)	11.6(1.1)	33.6(1.8)
RAG	82.1(1.4)	85.4(1.2)	28.4(2.5)	19.1(1.7)	10.7(1.3)	15.4(1.6)	40.2(1.7)
T-RAG	83.0(1.3)	85.0(1.2)	46.8(2.3)	37.9(1.9)	29.4(1.8)	22.1(2.0)	50.7(1.7)
Edit. KV	82.9(1.3)	83.9(1.5)	62.3(2.1)	50.9(2.0)	43.5(1.9)	38.0(2.2)	60.3(1.9)
Hybrid	84.1(1.2)	86.7(1.1)	65.0(1.9)	53.4(1.8)	46.1(1.8)	40.9(2.1)	62.7(1.8)
Graph	85.2(1.1)	87.0(1.0)	71.0(1.8)	63.5(1.7)	51.8(2.0)	45.4(2.3)	67.3(1.7)

Table 19: Per-family success (%) on the matched subset with Gemini 1.5 Pro 002. Cells are means (seed SD) over three ordering seeds.

Method	Store	Retr.	Rev.	Depr.	Erase	Prov.	Avg
No-Mem.	21.6(2.1)	19.2(2.2)	11.8(1.6)	8.4(1.3)	4.9(1.0)	3.1(0.8)	11.5(1.5)
Scratch.	60.5(2.0)	56.2(2.3)	30.4(2.4)	21.3(1.9)	13.8(1.5)	10.4(1.3)	32.1(1.8)
RAG	79.7(1.6)	82.8(1.4)	27.2(2.7)	18.4(1.9)	10.5(1.5)	14.2(1.8)	38.8(1.8)
T-RAG	80.4(1.4)	82.1(1.3)	44.8(2.4)	36.2(2.0)	27.9(1.9)	20.9(2.1)	48.7(1.8)
Edit. KV	80.2(1.5)	81.5(1.6)	60.3(2.3)	49.2(2.1)	41.7(2.0)	36.4(2.4)	58.2(1.9)
Hybrid	81.1(1.3)	83.4(1.2)	62.2(2.0)	50.8(1.9)	43.5(1.9)	37.8(2.2)	59.8(1.8)
Graph	82.8(1.3)	84.2(1.2)	68.8(2.0)	61.0(1.9)	49.8(2.2)	43.6(2.5)	65.0(1.8)

I Task Hardening and Manual Validation

Store/retrieve hardening inserts distractor sessions, rotates the queried alias, and screens out final sessions containing the answer string or a stemmed match. Only these task families were regenerated; the full 1,080-task pool was reevaluated, including its unchanged mutation tasks. No-Memory falls from 71.2/68.5 to 22.4/19.8 on store/retrieve. The structured backends retain their displayed scores, demonstrating that the change removes session-local shortcuts without suppressing genuine persistent-memory use.

I.1 Contract-fit and pass/fail audit

The 180-task audit samples 30 tasks per family and ten per tier within each family. Six external annotators independently assess whether the full episode requires the assigned operation and whether their pass/fail judgment agrees with the harness on a Graph Memory trajectory. Three annotators come from a US CS graduate program and three from a UK HCI group, with no overlap with the authors. They receive a common rubric, are paid above local minimum wage, and retain no identifying participant metadata. A senior annotator adjudicates cases flagged by at least two raters.

Contract-fit agreement is 174/180 (96.7%, Wilson CI [92.9,98.5]); harness/human pass/fail agreement is 171/180 (95.0%, CI [90.7,97.5]). The six contract-fit disagreements concern partial provenance

Table 20: Before/after hardening on store and retrieve with GPT-4o, three seeds. Mutation-family definitions are unchanged.

System	Store (pre)	Store (post)	Retrieve (pre)	Retrieve (post)
No-Memory	71.2	22.4	68.5	19.8
Scratchpad	74.8	62.8	72.1	58.4
RAG	82.3	82.3	85.7	85.7
Tombstone RAG	83.4	83.4	85.2	85.2
Editable KV	83.1	83.1	84.2	84.2
Hybrid RAG+KV	83.9	83.9	86.4	86.4
Graph Memory	85.6	85.6	87.3	87.3

chains. Krippendorff's nominal α is 0.83. These results validate the task semantics and the audited Graph trajectories as distinct objects.

Table 21: Instructions and adjudication rules for the manual audit. Annotators make both binary judgments and record a disagreement reason.

Judgment	Rule
Contract-fit	The task, read end-to-end, requires the nominally-assigned operation family; if a strictly easier family would suffice, mark MISLABEL.
Pass/fail match	The annotator's own pass/fail on the logged trajectory matches the harness label; disagreements are categorized as PARTIAL-PROVENANCE-ACCEPTED, RESIDUE-PARAPHRASE-DISPUTED, REVISION-CHAIN-TIE, or OTHER.
Adjudication	Cases flagged by ≥ 2 raters are resolved by a senior annotator blind to the harness outcome; the senior annotator's call is final.

On a separate 120-task slice (20 per family), two trained operators—one author and one external operator blind to the hypothesis—use full backend UI access, unrestricted scratchpad, rereads of prior sessions, and 20 minutes per task. The reported human-assisted macro is 93.8% [90.2,96.4]. This reference point measures performance with substantially greater inspection access than the budgeted agent and demonstrates substantial headroom above the model systems.

I.2 Five-family readout ablation

Excluding prove-origin leaves 900 task IDs across store, retrieve, revise, deprecate, and erase. Raw mutation obligations, including revise's audit-event check, remain in force. Table 22 therefore removes complete-chain readout while preserving bookkeeping obligations. The capability intervention further raises Hybrid's five-family macro to 70.4 against Graph's 72.0, reducing the reported gap to 1.6 points. The unchanged original-system gap after dropping prove-origin identifies obligations shared with the other families; the subsequent recovery after log/status exposure shows how to support them. Readout ablation, capability intervention, and interface tuning thus vary the scored obligation, accessible state, and presentation of that state, respectively.

Table 22: Original-system five-family macro after excluding prove-origin. The surviving 900 task IDs retain all other postconditions. Reported macros are rounded separately.

System	6-family macro (%)	5-family macro, no `prove-origin` (%)
No-Memory	11.9	13.7
Scratchpad	33.6	38.0
RAG	40.6	45.5
Tombstone RAG	51.1	56.8
Editable KV	60.6	65.0
Hybrid RAG+KV	62.4	66.8
Graph Memory	67.6	72.0

J External Evaluation: LongMemEval and LoCoMo

The external evaluations use 120 selected LongMemEval items and 100 selected LoCoMo items with GPT-4o. These are evaluation selections within the source resources, not official splits of those sizes. We preserve the source questions and conversational histories rather than inserting MemContract state transitions. LongMemEval's information-extraction, multi-session, temporal, knowledge-update, and abstention abilities provide a direct comparison with long-term question answering. Its single-session user, assistant, and preference question types can be grouped under information extraction [Wu et al., 2025]. LoCoMo supplies long conversational contexts and evaluates question answering and event-oriented understanding [Maharana et al., 2024]; an adversarial question in that resource is not, by itself, a policy-update task. We report overall external scores without relabeling its native categories as mutation contracts.

Table 23: External evaluation scores (%) for the original controlled systems. The two selected workloads retain their source objectives. Graph–Hybrid is 1.8 pp on LongMemEval and 2.8 pp on LoCoMo, compared with 5.2 on the balanced MemContract pool. Differences in both workload and scoring contribute to this attenuation.

System	LongMemEval (120 items)	LoCoMo (100 items)
No-Memory	24.9	18.4
Scratchpad	34.6	26.5
RAG	37.2	31.2
Tombstone RAG	46.5	38.6
Editable KV	49.2	41.9
Hybrid	52.4	44.3
Graph	54.2	47.1

The original controlled top-three ordering is preserved in these selected workloads. This is complementary to the contemporary-system finding in Table 1, where Zep/Graphiti leads. The external scores and the downstream correlations answer aggregate performance questions; the capability intervention identifies the specific state exposure that improves the contract outcome.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction identify the executable contracts, the audit-state mechanism interventions, and downstream correction handling, with the ten-system association cohort and correction-episode population distinguished. Tables 1, 3, and 4 distinguish contemporary systems, exposed capabilities, and interface tuning.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the task/domain scope, the empirical residue-probe boundary, the ten-system downstream association, and application-specific interpretation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper provides empirical evidence and explicit definitions. Appendix C gives the algebraic argument for positive-weight rank preservation and the assumptions for conditional scoring.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The common budgets, contract boundary, and statistical units are specified in Section 4 and Appendices A and C. Exact contemporary-system revisions, newer-model identifiers, and complete task-level records are not specified.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The submission specifies task construction, scoring, evaluation settings, and analysis definitions in the paper and technical appendices; it does not provide open access to code or data.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

   Justification: The common context, storage, cost, time, temperature, and seed settings are given in Section 4. The symmetric tuning comparison is described in Appendix D, but the tuning/evaluation split and full backend-specific configurations are not specified.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Table 1 reports macro confidence intervals and distinguishes rows with seed SDs. The capability and symmetric-tuning contrasts report paired intervals; Appendix C defines task clustering, template intervals, and domain sensitivity. Figure 2 shows uncertainty in system-level associations. External point estimates are identified as such.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Section 4 and Appendix C itemize five original studies and distinguish API inference from local processing. The added studies have common per-episode caps, but separate total compute usage is not reported.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: The benchmark uses constructed multi-session tasks. Expert annotation, compensation above local minimum wage, and handling of identifying metadata are described in Appendix I.1.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 7 explains how operation-level diagnosis supports reliable agents and why a passing fixed-probe erase audit must not be interpreted as a deletion certificate.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The evaluated memory interfaces operate on benchmark task content; this work does not introduce a pretrained generative model requiring a new model-release safeguard.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The evaluated systems and external benchmarks are cited at their use sites. The manuscript does not enumerate the license and access terms for each evaluated version.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper and appendices document the task population, contracts, coding rubric, and scoring boundary; an accompanying asset package is not part of this submission.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

Justification: Appendix I.1 describes annotator roles, the scoring rubric, adjudication, and compensation above local minimum wage. It does not reproduce the complete participant instructions or exact compensation amounts.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: Human contributions include expert annotation, coding, and auditing. An institutional approval or exemption determination is not documented in the paper.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are evaluated components of the memory-agent systems. Section 4 and Table 6 identify GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro 002, GPT-5.1, Claude Opus 4.5, and Gemini 3 Pro and their role in the protocol.