# StateLedger: Path-Addressed External Memory for Persistent Multi-Agent Systems

**Geer Yang**[1,2,3,5] **Bin Wu**[2] **Shaojiang Wang**[2,4,†] **Xixuan Liu**[2]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Institute of AI for Industries, Chinese Academy of Sciences, Nanjing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]Nanjing Institute of Software Technology, Nanjing, China
[5]Zhongke Yuanma (Chengdu) Service Robot Research Institute Co., Ltd., Chengdu, China

[†]Corresponding author: `seuhardware@aliyun.com`

## Abstract

Persistent LLM agents need memory that preserves state, provenance, validity, and evidence across long interaction histories. We introduce STATELEDGER, a path-addressed external memory substrate that stores memory as durable state evolution rather than flat retrievable items: events update canonical paths, registers absorb small changes, CommitPages record residual drift against Checkpoints, cue views bound traversal, and content-addressed artifacts avoid evidence duplication. We establish structural guarantees for storage, retrieval, coalescing, and conflict-aware ranking. Empirically, we evaluate STATELEDGER on LongMemEval, LoCoMo, and Evo-Memory Multi-turn, reaching 98.8% on LongMemEval-S, 82.6% on LongMemEval-M with a compressed durable footprint of 14.7% of raw session text, 95.8% LoCoMo non-adversarial accuracy, 87.1% overall success, and 93.5% overall progress on Evo-Memory.

## 1 Introduction

**Agents' memory as a systems problem.** Large language model (LLM) agents are increasingly deployed as persistent systems that span sessions, invoke tools, consume feedback, and coordinate with other agents, moving beyond stateless text generation. In this regime, memory is not merely a longer prompt: a practical substrate must decide *what to write*, *what to revise*, *what to keep private or shared*, *which evidence remains valid*, and *how to recover the right state* when future actions depend on past interactions. Recent evaluation settings sharpen this distinction: MemoryAgentBench decomposes agent memory into four complementary competencies, accurate retrieval, test-time learning, long-range understanding, and conflict resolution [Hu et al., 2025], while MemoryArena couples memorization with downstream action in multi-session Memory–Agent–Environment loops [He et al., 2026]. Together, these settings expose a storage and retrieval problem that flat vector search or rolling summaries do not adequately address.

**From interaction logs to evolving state.** Three observations motivate a dedicated external-memory design for multi-agent systems. *First, memory structure is intrinsically heterogeneous*: a single evaluation setting may mix factual retrieval, dynamic skill acquisition, long-range narrative abstraction, and overwrite/conflict handling, while deployment further crosses these with agent ownership, task phase, modality, and validity. *Second, generated histories are large*: MemoryAgentBench reports average inputs ranging from hundreds of thousands of tokens to well above one million tokens in recommendation-style test-time learning tasks [Hu et al., 2025]. *Third, event-level durable storage is redundant*: many consecutive interactions modify the same underlying state only marginally.

**Limitations of existing approaches.** Existing memory mechanisms meet these requirements only partially. History buffers preserve evidence but scale with event count; summary-based memory

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

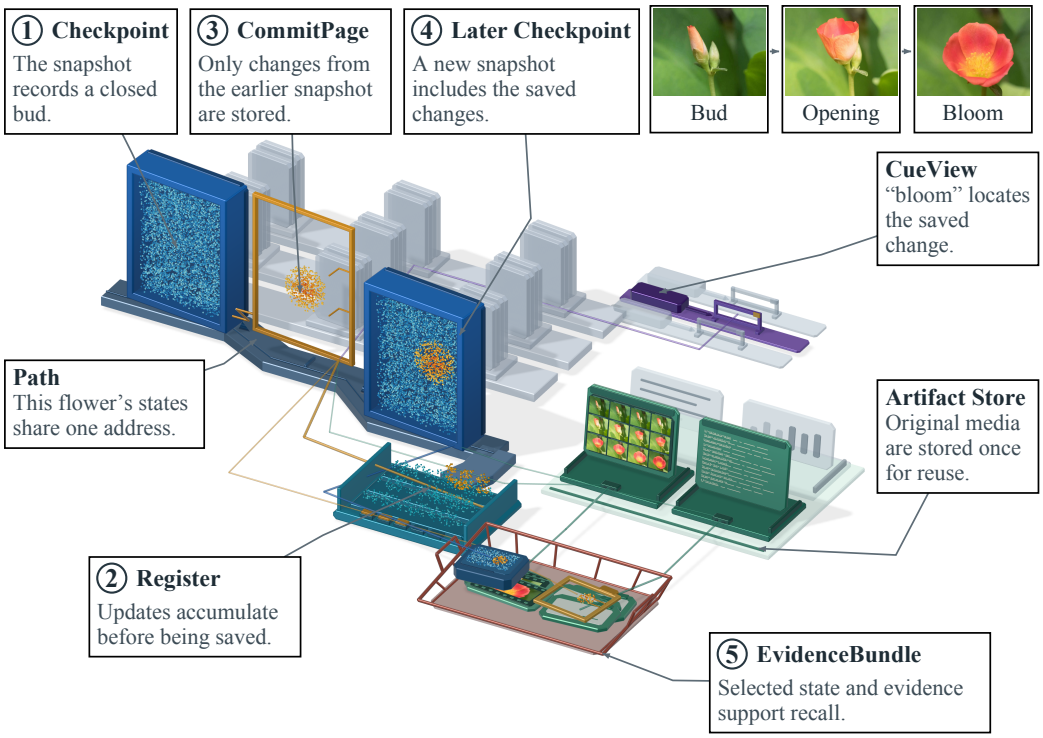


Figure 1: **StateLedger overview.** Circled numbers show successive stages along one state path: an earlier checkpoint, transient Register updates, a residual CommitPage, a later checkpoint, and evidence-supported recall. Raw media remain in a shared Artifact Store; CueViews provide traversal hints. Point clouds schematically depict state content and changes. Flower frames adapted from Ajith Samuel, CC BY 3.0.

reduces storage while weakening fine-grained provenance and update structure [Packer et al., 2023, Tan et al., 2025]; and BM25 or dense retrieval provide strong flat comparison methods [Robertson and Zaragoza, 2009, Karpukhin et al., 2020], but treat memory mainly as independent retrievable items rather than evolving external state. Graph- and hierarchy-based systems improve organization [Sun et al., 2026, Tang et al., 2026]. *Most memory systems optimize retrieval; StateLedger specifies what durable state remains after events.*

**StateLedger Memory.** We propose *StateLedger Memory*, a path-addressed external memory substrate for single- and multi-agent systems. Instead of storing memory as a bag of documents, StateLedger stores durable truth as a sequence of state updates. Each incoming event is resolved to a canonical state path. A per-path register absorbs small changes until the accumulated drift warrants a durable flush. The flush produces a *CommitPage*, namely a structured residual patch relative to a *Checkpoint*. Raw text, images, tool traces, and other evidence are stored once in a content-addressed artifact store. Durable coalescing merges compatible low-drift updates, so storage grows with meaningful state change rather than raw event frequency. At query time, StateLedger resolves candidate paths, reads checkpoints and local commit pages, optionally expands through bounded cue views, and fetches only those artifacts needed to assemble the final evidence bundle. Figure 1 gives an overview of StateLedger.

**Contributions.** StateLedger is designed for *external long-term memory*. It makes state structure explicit, keeps semantic truth in checkpoint/commit objects, and treats retrieval as state reconstruction rather than flat item search. Our contributions are fourfold: (i) we introduce a *path-ledger* abstraction that unifies state paths, checkpoints, residual commits, and content-addressed artifacts; (ii) we develop a durable coalescing mechanism that reduces persistent writes with answer preservation for state-sufficient queries; (iii) we design a path-local retrieval protocol with bounded cue expansion for efficient evidence assembly; and (iv) we present an empirical study on agent-memory bench-

marks showing that this design improves storage efficiency and task performance in settings that require retrieval, conflict resolution, and test-time memory updates.

## 2 Related Work

StateLedger lies at the intersection of *long-term agent memory*, *structured memory organization*, *durable state storage*, and *storage-aware retrieval*. We organize prior work around memory systems, structure, storage, retrieval, and evaluation settings.

**Agent and external memory.** LLM-agent systems have long treated memory as more than context extension: Generative Agents store observations, synthesize reflections, and retrieve memories for long-horizon behavior [Park et al., 2023]; Reflexion writes linguistic feedback into episodic memory for improvement across trials without weight updates [Shinn et al., 2023]; and MemGPT uses an operating-system analogy with virtual context management and tiered memory [Packer et al., 2023]. Recent external-memory systems operationalize this view: Mem0 extracts, consolidates, and retrieves salient memories [Chhikara et al., 2025]; MemOS treats memory as a first-class system resource with unified representation, scheduling, and lifecycle management [Li et al., 2025]; MIRIX extends persistent memory to multimodal and multi-agent settings through specialized stores [Wang and Chen, 2025]; and ByteRover lets the LLM curate a hierarchical Context Tree with relations, provenance, and lifecycle metadata [Nguyen et al., 2026]. For *write semantics*, StateLedger asks what durable external state remains after writes, revisions, conflicts, and evidence updates.

In the adjacent domain of virtual humans, Wang et al. [2025] survey generative models for talking-face and human-pose synthesis driven by visual, acoustic, and textual inputs.

**Structured, temporal, and hierarchical organization.** A second line of work imposes explicit structure over long-term interactions. RMM organizes utterance-, turn-, and session-level memories with reflective updates [Tan et al., 2025]; HiMem separates episodic and note memories and revises them through reconsolidation [Zhang et al., 2026]; Mnemis builds base and hierarchical graphs with dual-route retrieval [Tang et al., 2026]; and Chronos decomposes dialogue into temporally resolved event tuples indexed by event and turn calendars [Sen et al., 2026]. The key distinction is the *persistent unit*: StateLedger stores append-only residual updates on canonical state paths, consolidated by checkpointing and durable coalescing.

**Durable state, provenance, and storage systems.** StateLedger also draws on storage and database ideas: append-oriented logs, checkpointed recovery, content-addressed objects, provenance tracking, versioned state, and derived views. Event sourcing records state changes as reconstructable and auditable event sequences [Fowler, 2005]; LSM trees amortize high-rate writes through log-structured buffering and merge operations [O'Neil et al., 1996]; Venti identifies payloads by hash, supports write-once archival storage, and coalesces duplicate blocks [Quinlan and Dorward, 2002]; and database provenance studies why, how, and where data contributed to query results [Green et al., 2007, Cheney et al., 2009]. These storage principles are specialized to agent-produced writes and reads constrained by validity, provenance, and conflicts.

**Retrieval, indexing, and compression.** Retrieval-augmented generation and sparse or dense retrieval remain strong tools for memory access [Lewis et al., 2020, Robertson and Zaragoza, 2009, Karpukhin et al., 2020]. Efficient backends include FAISS-style dense retrieval [Johnson et al., 2021], HNSW [Malkov and Yashunin, 2020], product quantization [Jégou et al., 2011], and optimized product quantization [Ge et al., 2013]. These methods support efficient document retrieval and vector similarity search. StateLedger *retrieves* to reconstruct external state, rather than only to rank independent memory items.

**Evaluation settings for persistent memory.** Recent settings increasingly emphasize persistent and agentic memory. LoCoMo and LongMemEval study long-horizon conversational memory and multi-session reasoning [Maharana et al., 2024, Wu et al., 2025]; MemoryAgentBench evaluates incremental memory across accurate retrieval, conflict resolution, long-range understanding, and test-time learning [Hu et al., 2025]; and MemoryArena moves toward interdependent multi-session agentic loops, while Mem-Gallery extends long-term conversational memory to multimodal settings [He et al., 2026, Bei et al., 2026]. Our experiments combine long-horizon conversational memory with targeted settings requiring conflict handling, test-time updates, and downstream action.

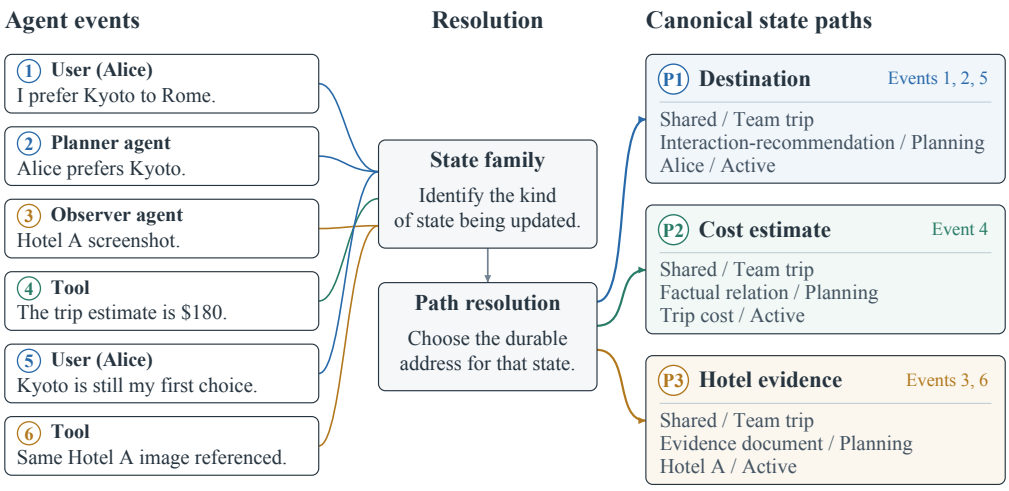


Figure 2: **Event-to-path resolution.** In a shared trip-planning example, events 1, 2, and 5 concern Alice's destination preference; event 4 concerns the trip-cost estimate; events 3 and 6 refer to the same Hotel A image. Addresses list scope, workspace, family, phase, entity group, and validity. The image resides in the Artifact Store. Routing selects an address; Register flushing determines durable writes.

## 3 Methodology

StateLedger Memory is an external long-term memory substrate for agents whose histories are not merely long, but *stateful*: facts are revised, skills are acquired at test time, evidence becomes stale, and multiple agents may write to shared or private state. Rather than store each event as an independent retrievable item, StateLedger stores memory as durable state evolution along canonical paths. Its central invariant is:

> *Durable semantic state lives only in Checkpoints and CommitPages.* Registers are transient write controllers, cue views are derived traversal hints, and artifacts store raw evidence rather than normalized state.

Thus StateLedger separates three concerns that are often entangled in flat memory systems: *what state is true*, *where raw evidence lives*, and *how future queries reach the relevant state*. Appendix B gives an end-to-end example, and Appendix A gives implementation details.

**Events, state families, and paths.** Let $u_{1:T}$ denote an event stream produced by users, tools, agents, and environment observations. Each event is first mapped to a typed state family and then resolved to a canonical state path:

$$f_t = \Phi(u_t), \qquad p_t = \Pi(u_t, f_t) \in \mathcal{P}. \tag{1}$$

Here $\Phi$ selects the *kind of state* being updated, while $\Pi$ selects the *durable address* at which that state should evolve. A path is therefore not a document identifier; it is a stable address for external state. In our implementation, a path is represented as a typed tuple,

$$p = (\text{scope}, \text{workspace}, \text{family}, \text{phase}, \text{entity_group}, \text{validity}). \tag{2}$$

The family field covers reusable memory modes, including factual-relation state, example-label state, interaction/recommendation state, narrative state, and evidence-document state. These families do not define separate memory stores. They parameterize one shared path-ledger substrate by determining path construction, register updates, coalescing policy, and readout preferences. Figure 2 illustrates this canonicalization step.

**Registers absorb small changes.** For each active path $p$, StateLedger maintains a typed in-memory register $h^p$ and an effective drift accumulator $a^p$. When event $u_t$ resolves to $p_t = p$, the path-local register is updated by a typed rule,

$$h_t^p = F_p(h_{t-1}^p, u_t), \qquad a_t^p \leftarrow a_{t-1}^p + \eta_t^p, \tag{3}$$

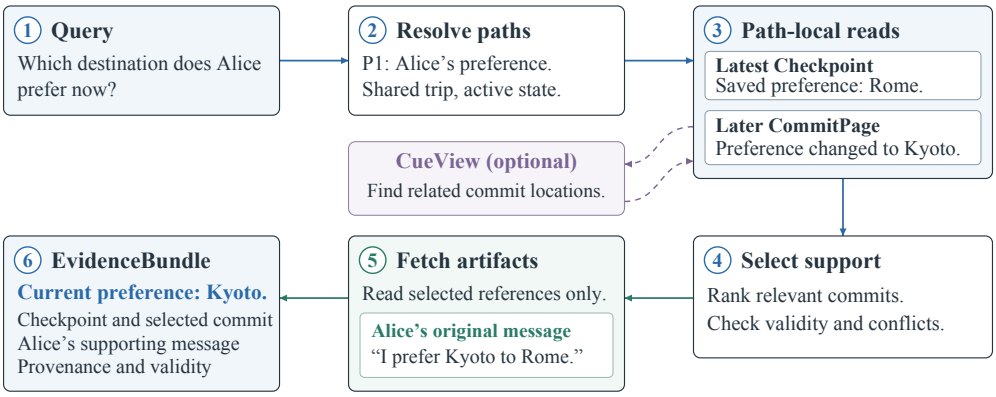


Figure 3: **Path-local retrieval and evidence reconstruction.** In this example, a checkpointed Rome preference is revised to Kyoto by a later CommitPage. The query resolves to P1, reads local state, and selects support before fetching Alice's original message. Optional cue expansion returns commit locations for further reads. The evidence bundle retains reconstructed state, selected objects, provenance, validity, and source evidence.

where $\eta_t^p \geq 0$ measures the effective state change induced by the event. Events assigned to other paths leave $h^p$ and $a^p$ unchanged. The register absorbs small or repetitive updates without immediately creating durable storage. A durable flush occurs when the accumulated drift reaches a path-specific threshold, $a_t^p \geq \tau_p$, or when a hard boundary occurs, such as a phase transition, conflict correction, handoff, explicit externalization event, or durable query feedback. This mechanism makes persistent growth depend on meaningful state change and required boundary events, rather than raw event frequency.

**Checkpoints and CommitPages.** A flush emits a *CommitPage* $\Delta_k^p$, a structured residual patch relative to a path-local *Checkpoint* $C_j^p$:

$$\Delta_k^p = \big(\mathrm{base}(C_j^p), \Delta^{\mathrm{slots}}, \Delta^{\mathrm{rels}}, \Delta^{\mathrm{art}}, \Delta^{\mathrm{valid}}, \Delta^{\mathrm{prov}}, \Delta h\big). \tag{4}$$

The slot and relation deltas, $\Delta^{\mathrm{slots}}$ and $\Delta^{\mathrm{rels}}$, record normalized state changes. The artifact delta, $\Delta^{\mathrm{art}}$, stores references to raw evidence payloads. The validity and provenance deltas, $\Delta^{\mathrm{valid}}$ and $\Delta^{\mathrm{prov}}$, record version, freshness, supersession, source, time, and author metadata. Finally, $\Delta h$ materializes the change in the typed register. A CommitPage is therefore not a copied transcript or document chunk. It is a compact state patch that can update or reconstruct path-local state when combined with its base checkpoint and referenced artifacts.

**Artifact externalization.** Raw evidence is stored separately in a content-addressed artifact store. Artifacts may contain text chunks, images, video frames, transcripts, tool traces, logs, or other payloads. CommitPages store only content hashes or references, so repeated evidence can be reused without duplicating payload bytes inside durable state updates. This design preserves provenance and exact evidence access while keeping the semantic ledger compact.

**Durable coalescing.** StateLedger reduces write amplification by coalescing compatible commit runs on the same path. The coalescing operator replaces a policy-selected, state-preserving contiguous sequence $\Delta_i^p, \ldots, \Delta_j^p$ with a reduced commit $\widetilde{\Delta}^p$. The reduced commit preserves latest-write slot semantics, reduced relation patches, required artifact-reference unions, reader-visible provenance, and validity/version order. The family policy compacts compatible runs in narrative, recommendation, and example streams while retaining exact atomic-evidence boundaries when required.

**Derived cue views.** A primary path gives StateLedger locality, but queries may mention entities, tasks, conflicts, or artifacts that cross path boundaries. To support bounded cross-path traversal, each CommitPage may emit a small number of cue signatures, such as entity cues, task cues, conflict-lineage cues, or artifact-lineage cues. A cue view is an append-only list of commit identifiers. It stores traversal hints, not semantic truth; if a cue view is lost, it can be rebuilt from Checkpoints, CommitPages, and artifact references.

**Path-local retrieval.** Given a query $q$, StateLedger first constructs a candidate path set $\mathcal{P}_q$ using query entities, state-family hints, temporal or validity constraints, cue signatures, and hot-path metadata. For each selected path, it reads the latest checkpoint, a bounded set of candidate CommitPages, and, when needed, a bounded cue-view window. Candidate commits are scored by path compatibility, family compatibility, evidence support, validity, and conflict status:

$$\begin{aligned} S(q, \Delta_k^p) =& \lambda_{\text{path}} s_{\text{path}}(q, p) + \lambda_{\text{family}} s_{\text{family}}(q, f_p) \\ &+ \lambda_{\text{evidence}} s_{\text{evidence}}(q, \Delta_k^p) + \lambda_{\text{valid}} s_{\text{valid}}(q, \Delta_k^p) + \lambda_{\text{conflict}} s_{\text{conflict}}(q, \Delta_k^p). \end{aligned} \tag{5}$$

The selected checkpoints, commits, cue expansions, provenance records, validity metadata, and fetched artifact snippets form an *evidence bundle*. Artifact payloads are fetched only after lightweight path and commit selection, so expensive evidence reads occur late in the pipeline.

**Unified evidence-supported readout.** The readout stage converts the evidence bundle into the answer requested by $q$ using a gpt-4o-mini API reader at temperature 0. Before the API call, StateLedger extracts and ranks candidate hints from state slots, relation patches, labels, entities, spans, narrative summaries, and artifact text by evidence support, path consistency, validity, provenance, conflict penalties, and answer-type compatibility. The gpt-4o-mini reader receives only the query, the ranked evidence bundle, and these candidate hints; it does not receive gold answers, hidden state, or benchmark-specific dispatch rules. This separates the storage problem from the answer-formatting problem: the ledger maintains durable state and assembles supported evidence, while the API reader formats the query-specific answer from that evidence. Figure 3 shows the retrieval protocol.

**Contrast with flat memory.** Flat memory systems store and retrieve independent items. StateLedger stores evolving external state. Its distinction is the *write–coalesce–retrieve–read* lifecycle: registers absorb small changes; CommitPages materialize residual patches; Checkpoints summarize stable path state; cue views enable bounded cross-path traversal; and artifacts preserve raw evidence without duplicating it inside every durable update. The resulting substrate is state-aware and hardware-conscious: writes are append-oriented, retrieval is path-local, and large artifact reads occur only after state reconstruction narrows the evidence set.

## 4 Theoretical Guarantees

**Theory roadmap.** The theory follows the cycle of a memory update in StateLedger. *(i) Write control:* Theorem 1, Corollary 1, and Theorem 2 show that durable CommitPage growth is charged to path-wise state change and forced boundaries, $H_p(T) + \lceil V_p(T)/\tau_p \rceil$. *(ii) Read locality:* Theorem 3 and Theorem 4 show that, after path resolution, a query reconstructs bounded path-local evidence—checkpoints, CommitPages, cue entries, and delayed artifacts—rather than scanning the full history. *(iii) Safe compaction:* Theorem 5 states when register absorption is exactly durable, and Theorem 7 with Corollary 4 proves that compatible CommitPage runs can be merged without changing any answer in $\mathcal{Q}_{\text{state}}$. *(iv) Conflict ranking:* Theorem 12 shows how path consistency, freshness, and stale penalties create a margin against stale-but-plausible supports over competing supports in the scored candidate set. *(v) Necessity and separation:* Theorems 8–11 explain why path addressing, validity metadata, shared artifact payloads, and absorption/coalescing are needed for these guarantees; Theorem 13 gives the resulting separation from flat event memory, and Theorem 14 connects these structural results to end-to-end correctness. Appendix C gives the formal model and complete proofs.

**Write-side storage.** For each path $p$, let $N_p(T)$ be the number of durable CommitPages emitted by time $T$, $V_p(T) = \sum_{t \le T : p_t = p} \eta_t^p$ the cumulative effective drift, $\tau_p > 0$ the flush threshold, and $H_p(T)$ the number of hard-boundary flushes. Theorem 1 proves

$$N_p(T) \le H_p(T) + \left\lceil \frac{V_p(T)}{\tau_p} \right\rceil. \tag{6}$$

Corollary 1 lifts this commit bound to a byte-level footprint bound over checkpoints, commits, cue views, artifact references, and unique content-addressed artifacts. Writing $\mathcal{C}_{\text{cp}}(T)$ for the total cost

of generating and writing Checkpoints, Theorem 2 gives the construction-time bound:

$$\mathcal{C}_{\mathrm{write}}(T) = O\Bigg( \sum_{t=1}^{T} \big[ \mathcal{C}_{\mathrm{parse}}(u_t) + \mathcal{C}_{\Phi}(x_t) + \mathcal{C}_{\Pi}(x_t, f_t) + D(|P_t|) + d_t \big] + \sum_{p \in P} \left( H_p(T) + \left\lceil \frac{V_p(T)}{\tau_p} \right\rceil \right) \bar{B}_p + \sum_{a \in U(T)} |a| + \mathcal{C}_{\mathrm{cp}}(T) \Bigg). \tag{7}$$

The per-path bound uses bounded CommitPage, cue-view, and artifact-reference sizes.

**Path-local retrieval.** For a query $q$, let $B(q)$ be the number of selected paths, $K(q)$ the CommitPage budget per path, $W(q)$ the cue-view budget per path, and $A(q)$ the number of fetched artifacts. Theorem 3 proves the object bound

$$R(q) \le c_{\mathrm{dir}} \log(1 + |P|) + B(q)\big(1 + K(q) + W(q)\big) + A(q). \tag{8}$$

Theorem 4 refines this into full query-time cost by adding query parsing, path-hypothesis generation, object decoding, scoring, artifact fetching, and evidence-bundle readout; in the experiments, the readout term is the gpt-4o-mini API reader call.

**Coalescing and durability.** Theorem 5 establishes the logging/flush requirement for exact durability and bounds the logical–durable gap between flushes by the drift threshold $\tau_p$. Theorem 7 constructs a concrete coalescing operator that preserves latest-write slots, reduced relations, artifact-reference unions, validity/version state, required provenance, and materialized register deltas. Corollary 4 then gives, for every $q \in \mathcal{Q}_{\mathrm{state}}$,

$$\mathrm{Ans}(q; C^p, \Delta_1^p, \ldots, \Delta_m^p) = \mathrm{Ans}(q; C^p, \Delta_1^p, \ldots, \Delta_{i-1}^p, \widetilde{\Delta}^p, \Delta_{j+1}^p, \ldots, \Delta_m^p). \tag{9}$$

**Validity-aware ranking.** For a fresh support $r$ and stale conflicting support $s$, Theorem 12 gives

$$\Pr[S_{\mathrm{SL}}(s) \ge S_{\mathrm{SL}}(r)] \le \exp\left( -\frac{(\mu_r - \mu_s + \alpha + \beta + \gamma)^2}{4\sigma^2} \right), \tag{10}$$

whenever the StateLedger margin $\mu_r - \mu_s + \alpha + \beta + \gamma$ is positive. The result covers the difficult case $\mu_s \ge \mu_r$, provided the path-consistency bonus, freshness bonus, and stale penalty overcome the flat-retrieval advantage of the stale support.

**Necessity and separation.** Theorems 8–11 show that the main components are forced by the target guarantees: path addressing for exact sublinear lookup, validity metadata for current-state conflicts, shared artifacts for repeated evidence, and absorption/coalescing for low-drift storage. On low-drift current-state workloads $\mathcal{W}_{n,m}$ with $n$ paths and $m$ events per path, Theorem 13 gives the storage separation

$$S_{\mathrm{SL}}(\mathcal{W}_{n,m}) \le S_{\mathrm{raw}}(\mathcal{W}_{n,m}) + O(n), \qquad S_{\mathrm{flat}}(\mathcal{W}_{n,m}) \ge S_{\mathrm{raw}}(\mathcal{W}_{n,m}) + \Omega(nm). \tag{11}$$

Theorem 14 combines these results into an end-to-end correctness guarantee under the conditions specified in Appendix C.8.

## 5 Experiments

**Experimental scope.** We evaluate StateLedger as a single path-addressed memory substrate across three regimes: long-term conversational QA, long conversational-memory QA, and multi-turn embodied or symbolic environments. We use the callable benchmark environment or evaluation harness associated with the corresponding benchmark when available. When a benchmark does not provide an official judge, the same gpt-4o-mini judge at temperature 0 is used. Details appear in Appendix D.

**LongMemEval.** *LongMemEval tests persistent multi-session dialogue memory.* We evaluate both S and M, which expands memory scale by nearly an order of magnitude and more directly stresses long-range state maintenance. StateLedger achieves *98.8%* on S and *82.6%* on M. Table 1 reports the question-type breakdown alongside local controls and published reference results. Table 2 shows the corresponding storage effect: longer histories yield stronger commit amortization, lower footprint/raw and read/raw ratios, and greater artifact reuse.

Table 1: LongMemEval S/M end-to-end QA accuracy.

| Method | Split | SSU | SSA | SSP | KU | TR | MS | Overall |
|---|---|---|---|---|---|---|---|---|
| StateLedger | S | **100.0** | 96.4 | **100.0** | **100.0** | **98.5** | **98.5** | **98.8** |
| Hindsight† [Latimer et al., 2025] | S | 97.1 | 96.4 | 80.0 | 94.9 | 91.0 | 87.2 | 91.4 |
| Mnemis† [Tang et al., 2026] | S | 98.6 | **100.0** | **100.0** | 93.6 | 86.5 | 86.5 | 91.6 |
| EMem-G† [Zhou and Han, 2025] | S | 94.8 | 87.5 | 50.0 | 94.4 | 83.7 | 82.6 | 84.9 |
| Mem0† [Chhikara et al., 2025] | S | 94.3 | 96.4 | 86.7 | 87.2 | 75.9 | 66.9 | 80.8 |
| BM25+TF-IDF hybrid | S | 94.3 | 89.3 | 40.0 | 84.6 | 36.8 | 46.6 | 61.0 |
| Flat dense | S | 94.3 | 87.5 | 33.3 | 84.6 | 36.8 | 42.9 | 59.4 |
| BM25 [Robertson and Zaragoza, 2009] | S | 92.9 | 91.1 | 36.7 | 78.2 | 35.3 | 42.1 | 58.2 |
| Rolling summary | S | 24.3 | 8.9 | 13.3 | 35.9 | 19.5 | 25.6 | 22.8 |
| History buffer | S | 11.4 | 0.0 | 6.7 | 11.5 | 5.3 | 9.0 | 7.6 |
| StateLedger | M | **95.7** | **91.1** | **53.3** | **92.3** | **78.9** | **76.7** | **82.6** |
| Round memory†, $K = V +$ fact, top-10 [Wu et al., 2025] | M | | | – | | | | 72.0 |
| Session memory†, $K = V +$ fact, top-5 [Wu et al., 2025] | M | | | – | | | | 71.4 |
| BM25 [Robertson and Zaragoza, 2009] | M | 90.0 | 87.5 | 16.7 | 78.2 | 30.8 | 33.8 | 52.8 |
| BM25+TF-IDF hybrid | M | 90.0 | 87.5 | 10.0 | 76.9 | 28.6 | 32.3 | 51.2 |
| Flat dense | M | 68.6 | 58.9 | 6.7 | 74.4 | 24.8 | 23.3 | 41.0 |
| Rolling summary | M | 20.0 | 8.9 | 6.7 | 35.9 | 18.0 | 19.5 | 19.8 |
| History buffer | M | 8.6 | 0.0 | 3.3 | 7.7 | 4.5 | 9.0 | 6.2 |

*Note.* SSU, SSA, SSP, KU, TR, and MS denote single-session user, single-session assistant, single-session preference, knowledge update, temporal reasoning, and multi-session. † denotes published reference values under their original evaluation settings (Appendix D.3).

Table 2: StateLedger storage and structural amortization.

| Benchmark / split | Events/Commit | Footprint/raw | StateIdx/raw | Read/raw | Artifact dedup. | Refs/artifact | Commits/path | Cues/Commit |
|---|---|---|---|---|---|---|---|---|
| LongMemEval-S | 33.95 | 41.1% | 3.94% | 1.51% | 22.6% | 1.29 | 1.54 | 3.00 |
| LongMemEval-M | **49.70** | **14.7%** | **1.23%** | **0.16%** | **79.4%** | **4.85** | **7.74** | 3.00 |

*Note.* The denominator is uncompressed rendered raw session text. Footprint includes ledger metadata and unique artifact payloads compressed with zlib level 6; it combines structural amortization, deduplication, and compression. StateIdx includes checkpoints, path directory, and cue views. Read/raw uses the mean top-10 retrieval bundle. Artifact dedup. is $1 - \#\text{unique}/\#\text{refs}$; Refs/artifact is $\#\text{refs}/\#\text{unique}$. Cues/Commit is the fixed cue-view budget.

**LoCoMo.** *LoCoMo complements LongMemEval with fine-grained reasoning over very long, multi-session personal dialogues.* While LongMemEval emphasizes larger-scale persistent memory, LoCoMo tests speaker-grounded fact retention, cross-session evidence linking, temporal ordering, and open-domain grounding. StateLedger obtains 95.8% non-adversarial judge accuracy (1,476/1,540).

Table 3: LoCoMo QA results (%).

| Method | Single | Multi | Temporal | Open | Overall |
|---|---|---|---|---|---|
| StateLedger | 96.3 | **97.5** | **95.0** | 89.6 | **95.8** |
| Mnemis $k = 30$ [Tang et al., 2026] | **97.1** | 92.9 | 90.7 | 79.2 | 93.9 |
| Mnemis, GPT-4.1-mini [Tang et al., 2026] | 96.2 | 91.8 | 90.3 | 82.3 | 93.3 |
| Mem0 token-efficient SDK [Mem0, 2026] | 92.3 | 93.3 | 92.8 | 76.0 | 91.6 |
| Hindsight [Latimer et al., 2025] | 86.2 | 70.8 | 83.8 | **95.1** | 89.6 |
| MemMachine [MemMachine, 2025] | 93.3 | 80.5 | 72.6 | 64.6 | 84.9 |
| Zep [Rasmussen et al., 2025] | 74.1 | 66.0 | 79.8 | 67.7 | 75.1 |
| TiMem [Li et al., 2026] | 81.4 | 62.2 | 77.6 | 52.1 | 75.3 |
| Mem0-Graph [Chhikara et al., 2025] | 65.7 | 47.2 | 58.1 | 75.7 | 68.4 |
| Mem0 [Chhikara et al., 2025] | 67.1 | 51.2 | 55.5 | 72.9 | 66.9 |

*Note.* Single, Multi, Temporal, and Open denote single-hop, multi-hop, temporal-reasoning, and open-domain QA.

**Evo-Memory Multi-turn.** *Evo-Memory Multi-turn evaluates memory in interactive task streams rather than retrospective dialogue QA.* Unlike LongMemEval and LoCoMo, which ask whether stored conversational state answers later questions, Evo-Memory tests whether an agent can update memory after each episode and reuse prior experience in embodied, instructional, symbolic-planning, and scientific environments. StateLedger yields 87.1% overall average success and 93.5% overall average progress with the environment adapters in Appendix D.5.

**Component diagnostics.** On LongMemEval, we analyze complete archived final-correct bundles (489 S and 385 M; Appendix D.6): *primary memories only* removes cue-driven candidate expansion, *metadata without artifact snippets* removes raw evidence text, and *snippets without JSON*

Table 4: Evo-Memory Multi-turn results.

| Method | ALFWorld | | BabyAI | | PDDL | | ScienceWorld | | Avg. | |
|---|---|---|---|---|---|---|---|---|---|---|
| | S | P | S | P | S | P | S | P | S | P |
| StateLedger | **100.0** | **100.0** | **75.0** | **83.4** | **100.0** | **100.0** | **73.3** | **90.5** | **87.1** | **93.5** |
| ReMem [Wei et al., 2025] | 92.0 | 96.0 | 73.0 | 83.0 | 83.0 | 95.0 | 62.0 | 89.0 | 78.0 | 91.0 |
| ExpRAG [Wei et al., 2025] | 74.0 | 89.0 | 62.0 | 72.0 | 72.0 | 89.0 | 46.0 | 76.0 | 63.0 | 82.0 |
| ExpRecent [Wei et al., 2025] | 66.0 | 83.0 | 63.0 | 73.0 | 53.0 | 76.0 | 49.0 | 82.0 | 58.0 | 79.0 |
| ReAct [Yao et al., 2023] | 51.0 | 75.0 | 57.0 | 72.0 | 75.0 | 91.0 | 44.0 | 77.0 | 57.0 | 79.0 |
| Mem0 [Chhikara et al., 2025] | 51.0 | 74.0 | 48.0 | 66.0 | 65.0 | 84.0 | 37.0 | 76.0 | 50.0 | 75.0 |
| DC-RS [Suzgun et al., 2025] | 50.0 | 74.0 | 52.0 | 68.0 | 62.0 | 84.0 | 34.0 | 74.0 | 50.0 | 75.0 |
| DC-Cu [Suzgun et al., 2025] | 50.0 | 74.0 | 50.0 | 67.0 | 62.0 | 84.0 | 33.0 | 75.0 | 49.0 | 75.0 |
| History [Wei et al., 2025] | 50.0 | 73.0 | 48.0 | 66.0 | 65.0 | 85.0 | 32.0 | 74.0 | 49.0 | 74.0 |
| SelfRAG [Asai et al., 2023] | 52.0 | 75.0 | 46.0 | 64.0 | 65.0 | 84.0 | 31.0 | 74.0 | 49.0 | 74.0 |
| AWM [Wang et al., 2024] | 49.0 | 73.0 | 53.0 | 68.0 | 60.0 | 82.0 | 34.0 | 74.0 | 49.0 | 74.0 |
| A-Mem [Xu et al., 2025] | 48.0 | 73.0 | 46.0 | 64.0 | 62.0 | 84.0 | 33.0 | 73.0 | 47.0 | 73.0 |
| No-memory comparison [Wei et al., 2025] | 18.0 | 49.0 | 51.0 | 66.0 | 17.0 | 39.0 | 10.0 | 53.0 | 24.0 | 52.0 |

*Note.* S and P denote task success and progress; Avg. is the overall mean across the four environments, with equal weight for each environment.

*metadata* removes explicit path, temporal, speaker, validity, and provenance fields. On LoCoMo, we reduce the retrieved context from top-50 to smaller top-$k$ budgets to test bounded cue/path expansion. Table 5 shows that candidate expansion has the largest coverage effect in LongMemEval-M, especially for multi-session cases; artifact snippets are most important on LongMemEval-S; structured metadata provides additional support beyond snippets; and LoCoMo evidence recall increases monotonically with the retrieval budget.

Table 5: Component diagnostics and retrieval-context controls.

| Benchmark / scope | Variant | Removed / control | $n$ | Diagnostic (%) | Drop | Breadth | Payload |
|---|---|---|---|---|---|---|---|
| LongMemEval-M | Full structured bundle | none | 385 | **90.5** | – | 29.9 cand. | 770.2 KB |
| LongMemEval-M | Primary memories only | cue/candidate expansion | 385 | 76.3 | -14.2 | 5.1 cand. | 66.5 KB |
| LongMemEval-M | Metadata only | artifact snippets | 385 | 88.1 | -2.5 | 29.9 cand. | 245.8 KB |
| LongMemEval-M | Snippets only | JSON state metadata | 385 | 88.6 | -1.9 | 29.9 cand. | 140.2 KB |
| LongMemEval-M MS | Full structured bundle | none | 90 | **79.9** | – | 29.8 cand. | 791.3 KB |
| LongMemEval-M MS | Primary memories only | cue/candidate expansion | 90 | 52.8 | -27.1 | 5.1 cand. | 69.2 KB |
| LongMemEval-M MS | Metadata only | artifact snippets | 90 | 75.2 | -4.7 | 29.8 cand. | 253.3 KB |
| LongMemEval-M MS | Snippets only | JSON state metadata | 90 | 75.6 | -4.3 | 29.8 cand. | 144.5 KB |
| LongMemEval-S | Full structured bundle | none | 489 | **90.0** | – | 11.7 cand. | 647.5 KB |
| LongMemEval-S | Primary memories only | cue/candidate expansion | 489 | 88.0 | -2.0 | 8.0 cand. | 227.8 KB |
| LongMemEval-S | Metadata only | artifact snippets | 489 | 78.6 | -11.4 | 11.7 cand. | 160.7 KB |
| LongMemEval-S | Snippets only | JSON state metadata | 489 | 89.2 | -0.8 | 11.7 cand. | 164.5 KB |
| LoCoMo all | Top-50 context | none | 1986 | **79.6** | – | 50 ctx. | 31.3 KB |
| LoCoMo all | Top-25 context | retrieval breadth | 1986 | 73.8 | -5.8 | 25 ctx. | 16.2 KB |
| LoCoMo all | Top-10 context | retrieval breadth | 1986 | 65.2 | -14.4 | 10 ctx. | 6.8 KB |
| LoCoMo all | Top-5 context | retrieval breadth | 1986 | 57.8 | -21.8 | 5 ctx. | 3.4 KB |
| LoCoMo all | Top-1 context | retrieval breadth | 1986 | 35.2 | -44.4 | 1 ctx. | 0.6 KB |

*Note.* Diagnostic (%) denotes pre-generation answer-token coverage for LongMemEval and full-evidence recall for LoCoMo. Payload is the mean serialized bundle or context size; Drop is relative to the full row within each scope.

**Reader portability and drift sensitivity.** Matched evaluations with three alternative readers yield token-F1 gains of 17.22–19.18 percentage points over each reader's strongest control (Appendix D.7). Across five replay seeds, raising the drift threshold from 0.10 to 0.35 reduces the durable commit ratio by 54.7% while maintaining 100% durable current-state accuracy (Table 21).

## 6 Limitations

StateLedger operates through a write/read interface that maps events to typed paths and retrieves query-relevant state and evidence. The formal query class consists of *state-supported* questions over durable path state, CommitPages, provenance, validity metadata, and fetched artifacts; language interpretation and abstraction are handled by the reader. The experiments cover long-term dialogue and multi-turn task memory. Adversarial-write robustness, privacy-policy enforcement, concurrent multi-writer consistency, and multimodal artifact fidelity are priorities for deployment evaluation. Low-drift and repetitive histories yield the largest storage gains; independent, query-relevant updates require proportionally more durable state.

## 7 Conclusion

We introduced StateLedger, a path-addressed external memory substrate that stores persistent agent memory as durable state evolution rather than flat retrievable items. Events resolve to canonical paths, registers absorb small changes, CommitPages record residual updates against Checkpoints, cue views support bounded traversal, and content-addressed artifacts preserve raw evidence without repeated payload storage. The theory shows drift-controlled commit growth, path-local retrieval, state-preserving coalescing, and validity-aware conflict ranking. The evaluations combine long-memory QA accuracy, multi-turn task success, and structural amortization measurements. Component diagnostics show that cue expansion, artifact snippets, structured metadata, and retrieval breadth each preserve answer-bearing evidence. These results frame long-term agent memory as a systems problem: deciding what external state should remain after many writes, revisions, conflicts, and artifact-producing actions.

## Acknowledgments

This work was supported by Nanjing Municipal Science and Technology Bureau (Grant No. 202512136), the Frontier Technologies R&D Program of Jiangsu (Grant No. BF2024052), the Department of Science and Technology of Jiangsu Province (Grant No. BM20250016), and Chengdu Science and Technology Program (Grant No. 2025-YF08-00097-GX).

The authors declare no additional competing interests.

## References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Yuanchen Bei, Tianxin Wei, Xuying Ning, Yanjun Zhao, Zhining Liu, Xiao Lin, Yada Zhu, Hendrik Hamann, Jingrui He, and Hanghang Tong. Mem-gallery: Benchmarking multimodal long-term conversational memory for MLLM agents. *arXiv preprint arXiv:2601.03515*, 2026. URL https://arxiv.org/abs/2601.03515.

James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009. doi: 10.1561/1900000006. URL https://doi.org/10.1561/1900000006.

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.

Martin Fowler. Event sourcing. https://martinfowler.com/eaaDev/EventSourcing.html, 2005. Accessed: 2026-05-04.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013. doi: 10.1109/CVPR.2013.379. URL https://doi.org/10.1109/CVPR.2013.379.

Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 31–40, New York, NY, USA, 2007. Association for Computing Machinery. doi: 10.1145/1265530.1265535. URL https://doi.org/10.1145/1265530.1265535.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026. URL https://arxiv.org/abs/2602.16313.

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/2507.05257v1.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57. URL https://doi.org/10.1109/TPAMI.2010.57.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL https://doi.org/10.1109/TBDATA.2019.2921572.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

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

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.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. TiMem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *arXiv preprint arXiv:2601.02845*, 2026. URL https://arxiv.org/abs/2601.02845.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, et al. Memos: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. URL https://arxiv.org/abs/2507.03724v1.

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. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.747/.

Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Mem0. The token-efficient memory algorithm now has temporal reasoning. https://mem0.ai/blog/the-token-efficient-memory-algorithm-now-has-temporal-reasoning, 2026. April baseline results; accessed November 6, 2026.

MemMachine. MemMachine reaches new heights on LoCoMo. Technical blog, 2025. URL https://memmachine.ai/blog/2025/09/memmachine-reaches-new-heights-on-locomo/. Accessed 2026-05-03.

Andy Nguyen, Danh Doan, Hoang Pham, Bao Ha, Dat Pham, Linh Nguyen, Hieu Nguyen, Thien Nguyen, Cuong Do, Phat Nguyen, and Toan Nguyen. ByteRover: Agent-native memory through LLM-curated hierarchical context. *arXiv preprint arXiv:2604.01599*, 2026. URL https://arxiv.org/abs/2604.01599.

Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O'Neil. The log-structured merge-tree (LSM-tree). *Acta Informatica*, 33(4):351–385, 1996. doi: 10.1007/s002360050048. URL https://doi.org/10.1007/s002360050048.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. 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*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Sean Quinlan and Sean Dorward. Venti: A new approach to archival storage. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, pages 89–101. USENIX Association, 2002. URL https://www.usenix.org/conference/fast-02/venti-new-approach-archival-data-storage.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 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*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Sahil Sen, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. Chronos: Temporal-aware conversational agents with structured event retrieval for long-term memory. *arXiv preprint arXiv:2603.16862*, 2026. URL https://arxiv.org/abs/2603.16862.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Haoran Sun, Shaoning Zeng, and Bob Zhang. H-MEM: Hierarchical memory for high-efficiency long-term reasoning in LLM agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 341–350, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.15. URL https://aclanthology.org/2026.eacl-long.15/.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025. URL https://arxiv.org/abs/2504.07952.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

Zihao Tang, Xin Yu, Ziyu Xiao, Zengxuan Wen, Zelin Li, Jiaxi Zhou, Hualei Wang, Haohua Wang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. Mnemis: Dual-route retrieval on hierarchical graphs for long-term LLM memory. *arXiv preprint arXiv:2602.15313*, 2026. URL https://arxiv.org/abs/2602.15313.

Pengcheng Wang, Haolun Peng, Siyang Mao, Shaojiang Wang, and Jie Tang. A review of generative models for virtual human motion driving. *Science China Information Sciences*, 68:181102, 2025. doi: 10.1007/s11432-023-4284-x. URL https://link.springer.com/article/10.1007/s11432-023-4284-x.

Yu Wang and Xi Chen. MIRIX: Multi-agent memory system for LLM-based agents. *arXiv preprint arXiv:2507.07957*, 2025. URL https://arxiv.org/abs/2507.07957.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-Memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025. URL https://arxiv.org/abs/2511.20857.

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://openreview.net/forum?id=pZiyCaVuti.

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. doi: 10.48550/arXiv.2502.12110. URL https://arxiv.org/abs/2502.12110.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Ningning Zhang, Xingxing Yang, Zhizhong Tan, Weiping Deng, and Wenyong Wang. Himem: Hierarchical long-term memory for LLM long-horizon agents. *arXiv preprint arXiv:2601.06377*, 2026. URL https://arxiv.org/abs/2601.06377.

Sizhe Zhou and Jiawei Han. A simple yet strong baseline for long-term conversational memory of llm agents, 2025. URL https://arxiv.org/abs/2511.17208.

# A Implementation Protocol

This section supplements the data model in Section 3 with write, read, and storage procedures.

**Addressing and invariants.** An event is normalized into text, entities, relations, artifact references, source, time, and metadata. A deterministic typed inducer selects its state family; the resolver then canonicalizes its scope, workspace, family, phase, entity group, and validity band into one path. Scope controls ownership and visibility, while the workspace and optional task namespace isolate unrelated histories. The family selects the register fields, coalescing policy, and readout preferences. Checkpoints and CommitPages hold normalized durable state; registers are volatile, artifacts hold raw evidence, and cue views are derived access structures. Durable writes append new objects rather than modifying prior state records in place.

**Write procedure.** For an event routed to path $p$:

1. Store each new raw payload under its content hash, together with modality, payload location, metadata, and access policy. Reuse the existing object for repeated content.
2. Update the typed register $h^p$ and accumulate effective drift, $a^p \leftarrow a^p + \eta_t^p$. Register fields include slots, relation keys, artifact references, temporal markers, and validity/provenance metadata.
3. Flush when $a^p \geq \tau_p$ or a hard boundary requires durability: a phase change, handoff, conflict correction, explicit externalization, query feedback, session close, or required artifact/provenance/permission update. Emit a CommitPage with its base-checkpoint identifier, slot and relation patches, artifact references, validity and provenance patches, and register delta; then reset $a^p$.
4. Append the commit location to the path directory and emit its configured cue signatures. Cue entries contain path/commit identifiers and offsets, not copied semantic payloads.

**Coalescing and checkpointing.** The family policy selects compatible contiguous commits on the same path for coalescing. Reduction preserves latest-write slots, composed relation updates, required artifact-reference unions, validity/version order, and reader-visible provenance. Atomic evidence remains separate when its exact boundaries are needed. A long chain, closing phase, or converged stable state can trigger a new Checkpoint. It folds the reduced state and required evidence references into a snapshot; older commits remain available as required by the provenance policy. Subsequent reconstruction starts from that checkpoint and applies only later state changes.

**Read procedure.**

1. Parse query entities, family hints, temporal/validity constraints, and answer type. Combine these with hot metadata to generate candidate paths.
2. Read their latest Checkpoints and candidate CommitPages. When needed, expand through bounded cue-view windows for entity, task, conflict, artifact, or source associations.
3. Rank candidates by the path, family, evidence, validity, and conflict terms defined in Section 3. Prefer active/latest supports for current-state queries; retain superseded supports when needed to explain lineage.
4. Fetch the selected artifact snippets through references in Checkpoints and CommitPages. Assemble checkpoint state, selected commits, validity, provenance, and snippets into the evidence bundle. Rank candidate answer hints from its slots, relations, spans, labels, and artifact text, then pass the query, bundle, and hints to the reader. The main QA experiments use gpt-4o-mini at temperature 0; alternative readers are evaluated in Appendix D.7.

**Storage layout and recovery.** Hot path directories, active registers, checkpoint pointers, access-control metadata, and small indices reside in CPU memory or a low-latency metadata service. Append-only Checkpoints, CommitPages, and cue records use durable SSD/NVMe or replicated KV/log storage. Large artifact payloads reside separately in object storage or a file system. Recovery replays path-local durable records and rebuilds cue views and other derived indices; replication can protect the ledger and artifact metadata across nodes. This separation keeps routing metadata close to the processor and defers large evidence reads until candidate selection.

## B Worked Example

Several agents investigate an HTTP 502 incident on `router-7`. The five events below update one factual-relation path in the shared `ops-prod` workspace, with incident phase, entity group `router-7`, and active validity. Let its initial checkpoint be $C^p_{17}$.

Table 6: Five source events produce three durable commits.

| Event | Observation or update | Durable action |
|---|---|---|
| $u_1$ | The monitor reports a 31% HTTP 502 rate and supplies a log. | Buffered in $h^p$. |
| $u_2$ | Bob proposes database saturation as the cause. | Buffered in $h^p$. |
| $u_3$ | Alice supplies a database trace with normal latency and no errors, rejecting Bob's hypothesis. | Conflict correction flushes $u_1$–$u_3$ as $\Delta^p_{18}$. |
| $u_4$ | Alice raises `upstream_timeout_s` from 10 to 30 and supplies the configuration record. | Externalization flushes $\Delta^p_{19}$. |
| $u_5$ | The monitor reports that the 502 rate is normal and the incident is resolved. | Resolution boundary flushes $\Delta^p_{20}$. |

**Absorption and durable state.** The first two updates stay below the drift threshold. Each subsequent boundary emits a CommitPage and resets the path's drift accumulator. $\Delta^p_{18}$ retains the 31% observation, Bob's claim, its rejection by Alice's trace, and their provenance. $\Delta^p_{19}$ records the timeout change and Alice's ownership; $\Delta^p_{20}$ records the normal metric and resolved status. Applying these three patches to $C^p_{17}$ yields the current state. The 31% value remains a historical observation rather than the current error rate.

**Evidence and access.** The log, trace, configuration record, and recovery metric are stored once as content-addressed artifacts. CommitPages hold their references. For example, the trace artifact records database p95 latency of 21 ms and zero errors; its full payload stays outside the ledger. Entity, conflict, and artifact cues point back to the relevant path-local commits. A conflict cue for database saturation reaches $\Delta^p_{18}$ without creating another copy of the claim.

**Checkpoint and query.** After the incident closes, a new checkpoint $C^p_{18}$ can fold in all three commits. It stores the 30-second timeout, normal 502 rate, resolved status, Alice's ownership, the superseded database hypothesis and rejection relation, and references to all four artifacts. Older commits remain available for lineage.

For the query "Why was Bob's database hypothesis rejected, and what fixed router-7?", retrieval selects this path and reads $C^p_{18}$. The evidence bundle includes the current state, the superseded claim with its rejecting evidence, and the selected trace/configuration artifacts fetched through checkpoint references. It may include $\Delta^p_{18}$–$\Delta^p_{20}$ for provenance, but does not apply them again: they are already folded into $C^p_{18}$. The reader can answer that the trace showed normal database latency and no errors, and that increasing the upstream timeout from 10 to 30 seconds resolved the incident.

## C Theory and Proofs

This appendix proves the storage, retrieval, coalescing, and conditional correctness results in Section 4.

### C.1 Objects and notation

**Definition 1** (Events, families, and paths)**.** For an event stream $u_{1:T}$, let $x_t = \mathrm{parse}(u_t)$, $f_t = \Phi(x_t)$, and $p_t = \Pi(x_t, f_t) \in P_t$. Here $P_t$ is the active path set and $P = \bigcup_{t \le T} P_t$. The shorthand $\Pi(u_t)$ denotes this parsing, family-induction, and path-resolution composition.

**Definition 2** (Registers, drift, and flushes)**.** An event assigned to $p$ updates its typed register by $h_t^p = F_p(h_{t-1}^p, x_t)$ and adds nonnegative drift $\eta_t^p$ to accumulator $a^p$. Other paths remain unchanged. Write $V_p(T) = \sum_{t \le T : p_t = p} \eta_t^p$. A threshold flush emits a CommitPage when $a^p \ge \tau_p > 0$; a hard-boundary flush emits one when $g_t^p = 1$, with $H_p(T) = \sum_{t \le T} g_t^p$. Hard boundaries include phase changes, handoffs, explicit externalization, conflict resolution, durable query feedback, and artifact or provenance boundaries. Simultaneous threshold and hard-boundary firing emits one CommitPage, counted as a hard-boundary flush. Every flush resets $a^p$ to zero.

**Definition 3** (Durable semantic objects)**.** A Checkpoint $C_j^p$ is a stable path-local base state. A CommitPage is the residual patch

$$\Delta_k^p = \big(\mathrm{base}(C_j^p), \Delta^{\mathrm{slots}}, \Delta^{\mathrm{rels}}, \Delta^{\mathrm{art}}, \Delta^{\mathrm{valid}}, \Delta^{\mathrm{prov}}, \Delta^{h}\big), \tag{12}$$

containing slot and relation updates, artifact references, validity/version/supersession metadata, provenance, and materialized register delta. Let $N_p(T)$ and $J_p(T)$ count emitted CommitPages and Checkpoints.

**Definition 4** (Artifacts and cue views)**.** The content-addressed artifact store materializes each payload in the unique-payload set $U(T)$ once; CommitPages retain references. A cue-view segment $V_{f,c} = [r_1, \ldots, r_m]$ contains CommitPage identifiers and optional path identifiers or offsets. These derived traversal structures carry no independent semantic truth.

### C.2 Storage and construction complexity

**Theorem 1** (Durable CommitPage bound)**.** *For every path $p$ and horizon $T$,*

$$N_p(T) \le H_p(T) + \left\lfloor \frac{V_p(T)}{\tau_p} \right\rfloor \le H_p(T) + \left\lceil \frac{V_p(T)}{\tau_p} \right\rceil. \tag{13}$$

*Proof.* Separate hard-boundary commits from threshold-only commits. There are at most $H_p(T)$ of the former. Each of the $m$ threshold-only commits consumes at least $\tau_p$ drift accumulated since the preceding flush. These intervals are disjoint because every flush resets the accumulator. Nonnegative drift therefore gives $m\tau_p \le V_p(T)$, and integrality gives the floor bound. The ceiling bound follows immediately. □

**Corollary 1** (Byte-level footprint bound)**.** *Let $B_{\mathrm{cp}}, B_{\mathrm{cm}}, B_{\mathrm{view}}, B_{\mathrm{ref}}$ bound the sizes of a Checkpoint, a CommitPage semantic patch excluding artifact references, a cue entry, and an artifact reference. If $M_{\mathrm{view}}(T)$ counts cue entries and $R_{\mathrm{art}}(T)$ counts references inside CommitPages, then*

$$\begin{aligned} S_{\mathrm{SL}}(T) \le \sum_{p \in P} J_p(T) B_{\mathrm{cp}} + \sum_{p \in P} \left( H_p(T) + \left\lfloor \frac{V_p(T)}{\tau_p} \right\rfloor \right) B_{\mathrm{cm}} \\ + M_{\mathrm{view}}(T) B_{\mathrm{view}} + R_{\mathrm{art}}(T) B_{\mathrm{ref}} + \sum_{a \in U(T)} |a|. \end{aligned} \tag{14}$$

*Proof.* Sum Checkpoint, CommitPage, cue-entry, reference, and unique-payload bytes. Substitute Theorem 1 for the CommitPage count; content addressing counts each payload once. □

**Corollary 2** (Strict savings over flat event logging under low state drift)**.** *Suppose $S_{\mathrm{flat}}(T) \ge b_{\mathrm{evt}} T$ for $b_{\mathrm{evt}} > 0$. If $\sum_p H_p(T) = o(T)$, $\sum_p V_p(T)/\tau_p = o(T)$, and all non-CommitPage terms in (14) are $o(T)$, then $S_{\mathrm{SL}}(T)/S_{\mathrm{flat}}(T) \to 0$. Constant-fraction assumptions give the corresponding constant-factor bound.*

*Proof.* Since $\lfloor x \rfloor \le x$, the CommitPage term is $o(T)$, as are the remaining terms. Divide $S_{\mathrm{SL}}(T) = o(T)$ by $S_{\mathrm{flat}}(T) = \Omega(T)$. The same substitution with explicit constant fractions gives the final claim. □

**Theorem 2** (Construction-time complexity)**.** *Let $\mathcal{F}_T$ index emitted CommitPages. For $(p,k) \in \mathcal{F}_T$, let $z_k^p$ count semantic patch atoms excluding artifact references, $r_k^p$ count artifact references, and $c_k^p = |\psi_{f(p)}(\Delta_k^p)|$ count emitted cue entries. Let $d_t$ be the register atoms touched by event $t$, $D(|P_t|)$ the directory cost, and $\mathcal{C}_{\mathrm{cp}}(T)$ the total cost of generating and writing Checkpoints. Define*

$$E_T = \sum_{t=1}^{T} \big[\mathcal{C}_{\mathrm{parse}}(u_t) + \mathcal{C}_{\Phi}(x_t) + \mathcal{C}_{\Pi}(x_t, f_t) + D(|P_t|) + d_t\big].$$

*Assume register updates cost $O(d_t)$, CommitPage serialization and indexing cost $O(z_k^p + r_k^p + c_k^p)$, and materializing a new payload costs $O(|a|)$. Assume an online path-local coalescer processes each emitted atom, reference, and cue entry $O(1)$ times amortized. Then*

$$\mathcal{C}_{\mathrm{write}}(T) = O\left(E_T + \sum_{(p,k)\in\mathcal{F}_T} (z_k^p + r_k^p + c_k^p) + \sum_{a\in U(T)} |a| + \mathcal{C}_{\mathrm{cp}}(T)\right). \tag{15}$$

*If $z_k^p + r_k^p + c_k^p \le \bar{B}_p$, then*

$$\begin{aligned}\mathcal{C}_{\mathrm{write}}(T) = O\Bigg(E_T + \sum_{p\in P} \left(H_p(T) + \left\lfloor \frac{V_p(T)}{\tau_p} \right\rfloor\right) \bar{B}_p \\ + \sum_{a\in U(T)} |a| + \mathcal{C}_{\mathrm{cp}}(T)\Bigg).\end{aligned} \tag{16}$$

*Balanced-tree or LSM-style directories give $D(|P_t|) = O(\log(1 + |P_t|))$; hash directories give expected $O(1)$ under standard expected-time assumptions.*

*Proof.* The per-event parsing, family induction, path resolution, directory access, and register updates sum to $E_T$. Serialization and indexing sum over emitted objects; amortized coalescing is charged to those same objects. Unique payloads are written once, and Checkpoint construction contributes $\mathcal{C}_{\mathrm{cp}}(T)$. This proves (15). For (16), bound the object sum by $\sum_p N_p(T)\bar{B}_p$ and apply Theorem 1. □

### C.3 Retrieval complexity

**Theorem 3** (Path-local retrieval object bound)**.** *Suppose query $q$ selects at most $B(q)$ paths, reads at most one Checkpoint, $K(q)$ CommitPages, and $W(q)$ cue entries per path, then fetches at most $A(q)$ artifacts. If directory lookup touches at most $c_{\mathrm{dir}} \log(1+|P|)$ objects, the total touched-object count satisfies*

$$R(q) \le c_{\mathrm{dir}} \log(1 + |P|) + B(q)(1 + K(q) + W(q)) + A(q). \tag{17}$$

*Proof.* Add the directory cost, the per-path object budget over at most $B(q)$ paths, and the final artifact budget. □

**Corollary 3** (Read-amplification bound)**.** *If Checkpoint, CommitPage, cue-entry, and fetched-artifact sizes are bounded by $B_{\mathrm{cp}}, B_{\mathrm{cm}}, B_{\mathrm{view}}, B_{\mathrm{art}}(q)$, respectively, bytes read satisfy*

$$Y(q) \le Y_{\mathrm{dir}}(q) + B(q)\big(B_{\mathrm{cp}} + K(q)B_{\mathrm{cm}} + W(q)B_{\mathrm{view}}\big) + A(q)B_{\mathrm{art}}(q). \tag{18}$$

*With all query-local budgets and object sizes bounded independently of $T$, object reads and bytes read depend on history length only through the directory term.*

*Proof.* Weight each object category in (17) by its maximum size. □

**Theorem 4** (Full query-time complexity)**.** *Let $P_q$ be the selected paths, $\mathcal{K}_p(q)$ and $\mathcal{W}_p(q)$ their selected CommitPages and cue entries, and $\mathcal{A}_q$ the fetched artifacts, under the preceding budgets. Let $\mathcal{C}_{\mathrm{qparse}}, \mathcal{C}_{\mathrm{hyp}}$ be query-parsing and path-hypothesis costs, and $\mathcal{C}_{\mathrm{cp}}, \mathcal{C}_{\mathrm{cm}}, \mathcal{C}_{\mathrm{view}}, \mathcal{C}_{\mathrm{art}}$ be object read/decode costs. Write $\mathcal{C}_S(q, \Delta)$ for the cost of the main-text score $S(q, \Delta)$, and $\mathcal{C}_G(q, \mathcal{E}_q)$ for evidence-bundle readout. With the preceding directory bound,*

$$\begin{aligned}\mathcal{C}_{\mathrm{read}}(q) = O\Bigg(&\mathcal{C}_{\mathrm{qparse}}(q) + \mathcal{C}_{\mathrm{hyp}}(q) + c_{\mathrm{dir}} \log(1 + |P|) \\ &+ \sum_{p \in P_q} \left[ \mathcal{C}_{\mathrm{cp}}(C^p_{\mathrm{latest}}) + \sum_{\Delta \in \mathcal{K}_p(q)} \big( \mathcal{C}_{\mathrm{cm}}(\Delta) + \mathcal{C}_S(q, \Delta) \big) + \sum_{v \in \mathcal{W}_p(q)} \mathcal{C}_{\mathrm{view}}(v) \right] \\ &+ \sum_{a \in \mathcal{A}_q} \mathcal{C}_{\mathrm{art}}(a) + \mathcal{C}_G(q, \mathcal{E}_q) \Bigg).\end{aligned} \tag{19}$$

*If object costs are bounded by $\bar{C}_{\mathrm{cp}}, \bar{C}_{\mathrm{cm}}, \bar{C}_{\mathrm{view}}, \bar{C}_{\mathrm{art}}$, and $\bar{C}_S(q)$ is the maximum candidate scoring cost (zero for no candidates), then*

$$\begin{aligned}\mathcal{C}_{\mathrm{read}}(q) = O\Big(&\mathcal{C}_{\mathrm{qparse}}(q) + \mathcal{C}_{\mathrm{hyp}}(q) + c_{\mathrm{dir}} \log(1 + |P|) \\ &+ B(q) \big[ \bar{C}_{\mathrm{cp}} + K(q)(\bar{C}_{\mathrm{cm}} + \bar{C}_S(q)) + W(q) \bar{C}_{\mathrm{view}} \big] + A(q) \bar{C}_{\mathrm{art}} + \mathcal{C}_G(q, \mathcal{E}_q) \Big).\end{aligned} \tag{20}$$

*Proof.* Sum parsing, hypothesis generation, directory access, path-local reads and scores, artifact fetching, and readout. Substituting the cardinality budgets and uniform costs gives the second bound. □

### C.4 Register absorption and exact durability

**Theorem 5** (Unflushed register gap and exact durability)**.** *Let $\mathsf{M}^p_t$ be the logical path state, including register effects, and $\widehat{\mathsf{M}}^p_t$ the state reconstructible from Checkpoints and CommitPages alone. Assume a pseudometric $D_p$ satisfies $D_p(\mathsf{M}^p_t, \mathsf{M}^p_{t-1}) \le \eta^p_t$ for events on $p$, while other events leave its state unchanged. Immediately after each flush, assume $\widehat{\mathsf{M}}^p_t = \mathsf{M}^p_t$. Between the last flush $s$ and time $t$,*

$$D_p(\mathsf{M}^p_t, \widehat{\mathsf{M}}^p_t) < \tau_p. \tag{21}$$

*Before the first flush take $s = 0$ and aligned initial states. If an acknowledged unflushed event changes a current-state answer but is persisted neither in a CommitPage nor a write-ahead register log, exact crash recovery for all current-state queries is impossible.*

*Proof.* The durable state remains $\widehat{\mathsf{M}}^p_t = \mathsf{M}^p_s$. The triangle inequality gives

$$D_p(\mathsf{M}^p_t, \widehat{\mathsf{M}}^p_t) \le \sum_{\ell=s+1}^{t:p_\ell=p} D_p(\mathsf{M}^p_\ell, \mathsf{M}^p_{\ell-1}) \le \sum_{\ell=s+1}^{t:p_\ell=p} \eta^p_\ell < \tau_p,$$

since otherwise a threshold flush would have occurred. For exact recovery, compare executions with and without an answer-changing volatile update. Their durable recovery inputs are identical, but their correct answers differ. No deterministic recovery procedure can answer both correctly; fixing a random tape gives the same conclusion for zero-error randomized recovery. □

### C.5 Coalescing correctness

**Definition 5** (Materialized path state and state-sufficient queries)**.** A materialized path state is $m = (L, R, A, V, \Gamma, H) \in \mathfrak{M}_p$: latest valid slots, reduced relations, artifact references, validity/version/supersession state, required provenance, and register state. Let $M_p(C^p, \Delta^p_{1:m})$ apply the CommitPages in order to a Checkpoint. A query belongs to $\mathcal{Q}_{\mathrm{state}}$ when its answer is a deterministic function $G_q$ of the relevant materialized states and artifacts selected from those states by $\mathcal{A}_q$. For one path,

$$\mathrm{Ans}(q; C^p, \Delta^p_{1:m}) = G_q\big(M_p(C^p, \Delta^p_{1:m}), \mathcal{A}_q(M_p(C^p, \Delta^p_{1:m}))\big). \tag{22}$$

For several paths, the maps receive their collection of states. Queries needing segmentation, verbatim wording, or audit traces belong to $\mathcal{Q}_{\text{state}}$ only when that information is retained in the state, provenance, or artifacts.

**Definition 6** (State-preserving coalescing)**.** A coalescer replaces a contiguous same-path run $R_{i:j} = (\Delta_i^p, \dots, \Delta_j^p)$ by one CommitPage $\widetilde{\Delta}^p = \mathsf{C}_p(R_{i:j})$. It is state-preserving when this substitution leaves $M_p$ unchanged for every prefix and suffix.

**Theorem 6** (Coalescing correctness)**.** *A state-preserving coalescer preserves the answer to every $q \in \mathcal{Q}_{\text{state}}$.*

*Proof.* Equal materialized states give equal selected artifacts under $\mathcal{A}_q$, hence identical inputs and outputs for $G_q$. □

**Definition 7** (Patch-induced transitions)**.** Each CommitPage induces a deterministic transition $T_\Delta : \mathfrak{M}_p \to \mathfrak{M}_p$:

$$\begin{aligned} T_\Delta(L, R, A, V, \Gamma, H) = \big(&L \oplus \Delta^{\text{slots}}, T^{\text{rel}}_{\Delta^{\text{rels}}}(R), A \cup \Delta^{\text{art}}, \\ &T^{\text{valid}}_{\Delta^{\text{valid}}}(V), T^{\text{prov}}_{\Delta^{\text{prov}}}(\Gamma), T^{h}_{\Delta^{h}}(H)\big). \end{aligned} \tag{23}$$

Here $\oplus$ overwrites precisely the supplied slot keys; the other transition maps are the family-specific materializer reducers. A run is representably compatible if its composed transition can be represented by one CommitPage of the same schema.

**Theorem 7** (Concrete state-preserving coalescer)**.** *For a representably compatible run $R_{i:j}$, construct $\widetilde{\Delta}^p$ by retaining, for each written slot $x$, the value in its last writer $\kappa(x) = \max\{k \in [i, j] : x \in \text{dom}(\Delta_k^{\text{slots}})\}$, and omitting unwritten slots. Set*

$$\begin{aligned} \widetilde{\Delta}^{\text{art}} &= \bigcup_{k=i}^{j} \Delta_k^{\text{art}}, \\ T^{\rho(b)}_{\widetilde{\Delta}^b} &= T^{\rho(b)}_{\Delta_j^b} \circ \cdots \circ T^{\rho(b)}_{\Delta_i^b}, \qquad b \in \{\text{rels}, \text{valid}, \text{prov}, h\}, \end{aligned} \tag{24}$$

*where $\rho(\text{rels}) = \text{rel}$ and $\rho(b) = b$ otherwise. Representable compatibility ensures these component patches exist. Assume $\Gamma$ contains exactly the provenance fields observable by $\mathcal{Q}_{\text{state}}$. Then, for every prefix state $m$,*

$$T_{\widetilde{\Delta}^p}(m) = T_{\Delta_j^p} \circ \cdots \circ T_{\Delta_i^p}(m), \tag{25}$$

*and the coalescer is state-preserving.*

*Proof.* For each slot, either the run leaves it unchanged or its final value is written by $\kappa(x)$; the coalesced patch produces the same result. Artifact references agree by associativity and idempotence of union. The relation, validity, provenance, and register components agree by their composed reducers in (24), including all query-observable provenance. Thus all six components agree for every prefix state. Applying the identical deterministic suffix to these equal states preserves equality of the final materialized states. □

**Corollary 4** (Concrete coalescing preserves state-sufficient answers)**.** *The construction in Theorem 7 preserves every answer in $\mathcal{Q}_{\text{state}}$.*

*Proof.* Apply Theorem 6 to the state-preserving construction. □

### C.6 Necessity results

**Theorem 8** (Necessity of a path directory for sublinear exact path lookup)**.** *Let a flat unordered layout contain $n$ cells, one pair $(p_i, b_i)$ per canonical path, with $b_i \in \{0, 1\}$, and no auxiliary path-to-cell directory or index. Any deterministic exact cell-probe algorithm answering all queries for $b_i$ must probe $n$ cells in the worst case. Thus $o(|P|)$ worst-case exact lookup requires a directory or equivalent index.*

*Proof.* An adversary returns non-target pairs for the first $n - 1$ distinct probes. If the reader stops earlier, place $(p_i, 0)$ or $(p_i, 1)$ in an unprobed cell and complete the other cells identically. The transcripts coincide but the correct answers differ, contradicting exactness. □

**Theorem 9** (Necessity of validity or version metadata for current-state conflicts)**.** *A representation that stores only an unordered support set, with no time, version, validity, supersession, overwrite, or equivalent order information, cannot support deterministic exact latest-valid-value answers for all event orders.*

*Proof.* Histories $(x \leftarrow 0, x \leftarrow 1)$ and $(x \leftarrow 1, x \leftarrow 0)$ both yield $\{x = 0, x = 1\}$, but require answers 1 and 0. Identical reader inputs cannot produce both correct answers. □

**Theorem 10** (Storage advantage and necessity of shared artifact payloads)**.** *Let a payload have size $L$ after any admissible codec and occur in $m$ CommitPages. Full copying costs $S_{\mathrm{copy}}(m) \geq mL$, whereas one shared payload with references of size $b_{\mathrm{ref}}$ costs $S_{\mathrm{ca}}(m) \leq L + mb_{\mathrm{ref}}$. Consequently,*

$$\frac{S_{\mathrm{ca}}(m)}{S_{\mathrm{copy}}(m)} \leq \frac{1}{m} + \frac{b_{\mathrm{ref}}}{L}. \tag{26}$$

*Preserving full raw evidence with $o(mL)$ storage requires shared payloads, references, deduplication, dictionary coding, or an equivalent mechanism.*

*Proof.* Count $m$ independent copies versus one payload and $m$ references. Without sharing, each independent decodable representation costs at least the post-codec size $L$, giving $\Omega(mL)$ bytes and contradicting $o(mL)$. □

**Theorem 11** (Necessity of absorption or coalescing for low-drift semantic storage)**.** *Suppose $m$ events on one path determine a bounded-size state-sufficient representation, with $V_p(m) = O(1)$, $H_p(m) = O(1)$, fixed $\tau_p > 0$, and bounded CommitPage size. Then* StateLedger *emits $O(1)$ CommitPages. An immutable log retaining one non-coalescible semantic record of at least $b_{\mathrm{evt}} > 0$ bytes per event uses $\Omega(m)$ bytes. Exact state-sufficient memory with $o(m)$ semantic storage must instead absorb, coalesce, delete, rewrite into aggregates, or equivalently avoid immutable per-event storage.*

*Proof.* Theorem 1 gives $N_p(m) \leq H_p(m) + \lfloor V_p(m)/\tau_p \rfloor = O(1)$. Bounded Checkpoint and metadata overhead then gives bounded semantic storage. The immutable log requires at least $mb_{\mathrm{evt}}$ bytes; any sublinear representation must avoid retaining those independent records. □

### C.7 Conflict-aware ranking

**Lemma 1** (Sub-Gaussian difference)**.** *Let $\xi_r, \xi_s$ be independent mean-zero $\sigma$-sub-Gaussian variables, with $\sigma > 0$: $\mathbb{E}e^{\lambda\xi_i} \leq e^{\lambda^2\sigma^2/2}$ for all $\lambda \in \mathbb{R}$. Then $\xi_s - \xi_r$ is mean-zero $\sqrt{2}\sigma$-sub-Gaussian, and*

$$\Pr[\xi_s - \xi_r \geq t] \leq \exp\left(-\frac{t^2}{4\sigma^2}\right), \qquad t > 0. \tag{27}$$

*Proof.* Independence gives $\mathbb{E}e^{\lambda(\xi_s - \xi_r)} = \mathbb{E}e^{\lambda\xi_s}\mathbb{E}e^{-\lambda\xi_r} \leq e^{\lambda^2\sigma^2}$. Chernoff's bound is $e^{-\lambda t + \lambda^2\sigma^2}$; choosing $\lambda = t/(2\sigma^2)$ proves the tail inequality. □

**Theorem 12** (General conflict-margin ranking bound)**.** *For fresh correct support $r$ and stale conflicting support $s$ on one key, let $S_{\mathrm{flat}}(i) = \mu_i + \xi_i$, with independent mean-zero $\sigma$-sub-Gaussian noise and $\sigma > 0$. Let*

$$S_{\mathrm{SL}}(r) = \mu_r + \alpha + \beta + \xi_r, \qquad S_{\mathrm{SL}}(s) = \mu_s - \gamma + \xi_s,$$

*where $\alpha, \beta, \gamma \geq 0$ are path-consistency and validity/freshness bonuses and a stale penalty. If $\delta_{\mathrm{SL}} = \mu_r - \mu_s + \alpha + \beta + \gamma > 0$, then*

$$\Pr[S_{\mathrm{SL}}(s) \geq S_{\mathrm{SL}}(r)] \leq \exp\left(-\frac{\delta_{\mathrm{SL}}^2}{4\sigma^2}\right). \tag{28}$$

*This also applies when $\mu_s \geq \mu_r$, provided $\alpha + \beta + \gamma > \mu_s - \mu_r$.*

*Proof.* The stale-first event is exactly $\xi_s - \xi_r \geq \delta_{\mathrm{SL}}$. Apply Lemma 1 at this positive threshold. □

### C.8 Separation from flat memory and conditional correctness

**Theorem 13** (Separation from flat event memory on low-drift current-state workloads)**.** *Consider $\mathcal{W}_{n,m}$: $n = |P|$ paths receiving $m \geq 1$ events each about one current-state key. Every path flushes pending updates at its final event, counting this terminal flush in $H_p(T)$. Assume, per path: (i) a state-sufficient final representation of size $O(1)$; (ii) $H_p(T) = O(1)$ and $V_p(T)/\tau_p = O(1)$; (iii) $O(1)$ Checkpoint, CommitPage, cue-view, artifact-reference, and required-provenance overhead; and (iv) a current-value query $q_i$ that reads its latest Checkpoint and all subsequent CommitPages, with $B(q_i) = 1$ and $K(q_i), W(q_i), A(q_i) = O(1)$. If $S_{\mathrm{raw}}$ is the cost of required unique raw payloads, then*

$$S_{\mathrm{SL}}(\mathcal{W}_{n,m}) \leq S_{\mathrm{raw}}(\mathcal{W}_{n,m}) + O(n), \qquad R(q_i) = O(\log(1+n)+1). \tag{29}$$

*A flat memory retaining each event as an independent semantic item of at least $b_{\mathrm{evt}} > 0$ bytes uses $S_{\mathrm{flat}} \geq S_{\mathrm{raw}} + \Omega(nm)$ when preserving the same raw payloads. If it has no path directory, no validity/version index, and no per-path current-state reducer, worst-case exact answering requires $\Omega(n)$ probes to locate an arbitrary path. Even with that path granted, it requires $\Omega(m)$ inspected candidates if its $m$ items are indistinguishable to the retriever up to adversarial tie-breaking unless inspected.*

*Proof.* The terminal flush adds at most one hard boundary per path. Theorem 1 gives $N_p(T) = O(1)$; bounded per-path overhead therefore totals $O(n)$, plus $S_{\mathrm{raw}}$. Terminal flushing makes the final state durable, and the latest Checkpoint and subsequent CommitPages reconstruct it. The budgets and Theorem 3 yield the read bound.

The flat layout stores $nm$ records, costing at least $b_{\mathrm{evt}}nm$ in addition to raw payloads. The path-location lower bound follows from Theorem 8. On a granted path, any reader inspecting fewer than $m$ candidates leaves one unseen. An adversary can put the latest valid update there, with value 0 or 1 in two histories agreeing on every inspected item. With neither an order index nor a reducer, the transcripts coincide while the correct values differ. Thus exact worst-case answering requires $\Omega(m)$ inspections. □

**Theorem 14** (Conditional end-to-end correctness for state-sufficient queries)**.** *For $q \in \mathcal{Q}_{\mathrm{state}}$ at time $T$, let $P^\star(q)$ be the paths whose states and artifacts determine answer $y^\star(q)$. Assume:*

1. ***Exact durability:*** *$\widehat{\mathsf{M}}^p_T = \mathsf{M}^p_T$ for every $p \in P^\star(q)$.*
2. ***Path recall:*** *$P^\star(q) \subseteq P_q$.*
3. ***Budget coverage:*** *the selected Checkpoints, CommitPages, cue entries, and artifacts reconstruct every state component and artifact required by $q$.*
4. ***State-preserving coalescing:*** *every coalesced run on a required path satisfies Theorem 7.*
5. ***Conflict coverage and margins:*** *every required fresh–stale pair $(r,s) \in \mathcal{C}(q)$ enters the scored candidates, with $\delta_{r,s} = \mu_r - \mu_s + \alpha_{r,s} + \beta_{r,s} + \gamma_{r,s} > 0$.*
6. ***Reader consistency:*** *when the bundle contains the unique state-supported answer with required validity and provenance, the reader returns $y^\star(q)$.*

*Under the ranking model of Theorem 12,*

$$\Pr[\widehat{\mathrm{Ans}}(q) = y^\star(q)] \geq 1 - \sum_{(r,s)\in\mathcal{C}(q)} \exp\left(-\frac{\delta^2_{r,s}}{4\sigma^2}\right). \tag{30}$$

*If $\mathcal{C}(q) = \emptyset$, or scoring is deterministic and every required fresh support outranks its stale competitors, correctness holds with probability 1.*

*Proof.* Durability, path recall, and budget coverage recover every required state component and artifact. Theorem 7 and Corollary 4 preserve their state-sufficient answers through coalescing. For each competing pair, Theorem 12 bounds stale-first probability by $e^{-\delta^2_{r,s}/(4\sigma^2)}$. A union bound over $\mathcal{C}(q)$ gives (30): outside these events, all required fresh supports are available, and reader consistency returns $y^\star(q)$. With no conflicts or deterministic correct ordering, no stale-first event occurs. □

# D Experimental Details

This appendix gives the benchmark adapters, execution settings, and supporting results for Section 5.

## D.1 StateLedger implementation

**Events and artifacts.** All benchmarks use the write/read protocol in Appendix A. Dialogue adapters emit one SourceEvent per non-empty utterance, retaining sample, session, turn, speaker, timestamp, text, and source identifiers. Interactive adapters emit one event per environment transition, retaining episode, step, observation, action, feedback, termination flag, and source identifier. Events follow native chronological order and preserve session/episode boundaries. Gold answers, answer-session identifiers, hidden simulator state, expert trajectories, and scoring labels enter only post-prediction evaluation. Raw payloads are serialized as canonical UTF-8 JSON and content-addressed by SHA-256; repeated payloads share one artifact, referenced by hash and source span.

**Paths and durable state.** The path prefix isolates workspace and sample/episode; its remaining fields encode state family, phase, actor/entity, scope, and validity. Dialogue adapters extract speaker, entity, time, and preference cues; interactive adapters extract task phase, object, predicate, goal, and action-outcome cues. Unresolved fields use a deterministic unknown value. Register slots are lower-cased and normalized; timestamps retain their native representation and an order index. The benchmark fixes its drift and hard-boundary policy before scoring. Checkpoints retain active slots, reduced relations, validity/provenance, commit pointers, and artifact references. Compatible runs use the coalescer in Appendix C.5. Cue signatures project path fields, entities, temporal anchors, relations, artifacts, and task/action terms into bounded commit-id lists; LongMemEval uses up to three signatures per commit.

**Evidence and readout.** Routing combines path/family compatibility, lexical and cue overlap, validity, freshness, and provenance. The evidence bundle separately records memory/path identifiers, family/phase, speaker or actor, entity, time/session/turn, validity, checkpoint state, commit changes, conflict status, artifact snippets, provenance, and score components. The main QA reader is gpt-4o-mini at temperature 0. Its prompt requests concise evidence-supported answers, active/latest-valid support for current-state questions, and abstention when evidence is insufficient. Tables 8, 12, and 14 give the benchmark-specific budgets and controllers; Appendix D.7 reports alternative readers.

**Reproduction materials.** The benchmark protocols below specify data preparation, retrieval and context budgets, reader and judge settings, and environment interfaces. Benchmark-provided prompt conventions are followed where applicable, together with the evidence-supported readout described above. Code availability appears in Appendix D.8.

**Measurement.** Predictions are saved as JSONL with query and evidence ids, bundle hash, model/decoding settings, raw and normalized responses, and final score. Storage and retrieval measurements use the same frozen construction as prediction. Durable storage includes ledger metadata and unique artifact payloads; query-time bundle bytes are reported separately.

## D.2 Statistical reporting and compute resources

**Statistical reporting.** Table 7 reports exact correct counts and Wilson 95% binomial intervals for binary metrics on the fixed benchmark splits. With $\hat{p} = x/n$ and $z = 1.96$, the interval is

$$\frac{\hat{p} + z^2/(2n) \;\pm\; z\sqrt{\hat{p}(1-\hat{p})/n + z^2/(4n^2)}}{1 + z^2/n}.$$

These intervals quantify finite-sample uncertainty for the recorded binary outcomes on each fixed benchmark split. Progress, storage, retrieval, and latency diagnostics are aggregates over the instances counted in the corresponding tables.

**Compute resources.** Experiments ran on a Windows 11 machine with an AMD Ryzen AI 9 HX 370 CPU (12 cores, 24 threads), 32 GB RAM, and an NVIDIA GeForce RTX 5060 Laptop GPU. StateLedger construction, retrieval, bundle organization, and environment interaction use local CPU

Table 7: Finite-sample uncertainty for main binary metrics.

| Metric | Correct / $n$ | Score (%) | Wilson 95% CI (%) |
|---|---|---|---|
| LongMemEval-S QA accuracy | 494 / 500 | 98.8 | [97.4, 99.4] |
| LongMemEval-M QA accuracy | 413 / 500 | 82.6 | [79.0, 85.7] |
| LoCoMo non-adversarial judge accuracy | 1,476 / 1,540 | 95.8 | [94.7, 96.7] |
| LoCoMo all-category judge accuracy | 1,922 / 1,986 | 96.8 | [95.9, 97.5] |
| Evo-Memory Multi-turn episode-weighted success | 344 / 396 | 86.9 | [83.2, 89.8] |

processing; model inference primarily uses external APIs. No model training or fine-tuning is performed. LoCoMo uses sentence-transformers/all-MiniLM-L6-v2 (embedding batch size 32) where dense features are enabled. A completed evaluation round took approximately 10 days of elapsed time; the overall experimental effort took approximately 20 days. These are end-to-end elapsed times, including API calls. Per-query and per-episode local timings and storage footprints appear in the benchmark tables.

### D.3 LongMemEval details

Both LongMemEval-S and LongMemEval-M contain 500 questions: SSU 70, SSA 56, SSP 30, KU 78, TR 133, and MS 133. Category accuracies appear in the main results. The construction stage stores full sessions as content-addressed artifacts, converts non-empty dialogue turns into SourceEvents, and runs them through the shared path/register/flush pipeline to produce CommitPages and checkpoints. The quality tables report question-level accuracy for the final StateLedger implementation; storage metrics describe the evaluated construction.

**Published references.** The dagger-marked rows in Table 1 are literature results. Hindsight values are from Table 3 of Latimer et al. [2025], with a Gemini-3 Pro reader and GPT-OSS-120B memory and judge. The Mnemis, EMem-G, and Mem0 S values are from the GPT-4.1-mini block of Table 2 of Tang et al. [2026]; method citations identify the original systems. The M round/session-memory values are from Wu et al. [2025]. Published rows preserve their original category and overall values and aggregation conventions. The unmarked retrieval, summary, and history rows are local controls.

Table 8: LongMemEval execution protocol.

| Stage | Role | Setting |
|---|---|---|
| Stage 0 | data/split validation | validate question ids, sessions, date metadata, type distribution |
| Stage 1 | StateLedger construction | non-empty dialogue turns become SourceEvents; full sessions are content-addressed artifacts; paths use scope/workspace/family/phase/speaker/entity/validity |
| Stage 2 | path/state routing | top_k=50, path_budget=12, local_depth=8, dense_rerank=false |
| Stage 3 | evidence retrieval | session_pool=30, top_k=20, window_radius=1, per_session_cap=3 |
| Stage 4 | bundle organization | top_k=10, primary_k=2, rehydrate_window_radius=4 |
| Stage 5 | reader and judge | reader=gpt-4o-mini, judge=gpt-4o-mini, max_memories=10, max_candidates=20, max_chars_per_memory=5000, reader_max_tokens=800, judge_max_tokens=12, temperature=0 |

Table 9: LongMemEval split-level storage. Footprints include JSONL ledger metadata and unique artifact payloads compressed with zlib level 6; raw input is uncompressed rendered session text.

| Metric | LongMemEval-S | LongMemEval-M |
|---|---|---|
| SourceEvents | 246,738 | 2,446,698 |
| Sessions | 23,867 | 237,655 |
| CommitPages | 7,268 | 49,233 |
| Checkpoints | 4,722 | 6,364 |
| Paths | 4,722 | 6,364 |
| Cue entries | 21,789 | 147,671 |
| Artifact references | 23,867 | 237,655 |
| Unique artifacts | 18,464 | 49,018 |
| Durable footprint (MB) | 102.8 | 364.5 |
| Raw input (MB) | 250.4 | 2475.6 |
| Footprint / raw (%) | 41.1 | 14.7 |
| Events / CommitPage | 33.95 | 49.70 |

Table 10: LongMemEval Stage 3 retrieval diagnostics by split.

| Metric | LongMemEval-S | LongMemEval-M |
|---|---|---|
| AS@1 (%) | 87.6 | 68.6 |
| AS@5 (%) | 98.0 | 86.6 |
| AS@10 (%) | 99.4 | 91.8 |
| Sup@1 | 89.2 | 74.4 |
| Sup@5 | 98.0 | 90.0 |
| Sup@10 | 99.4 | 93.4 |
| EvR@1 | 56.6 | 45.5 |
| EvR@5 | 91.6 | 76.2 |
| EvR@10 | 96.2 | 82.9 |
| Gold@10 | 49.6 | 49.6 |
| MRR-AS | 0.921 | 0.765 |
| Seconds / query | 0.588 | 4.107 |

### D.4 LoCoMo details

LoCoMo is evaluated as full QA over five categories. The benchmark category mapping is 1=multi-hop, 2=temporal, 3=open-domain, 4=single-hop, and 5=adversarial. Table 11 reports integer correct counts so the aggregate and category results can be reconciled directly.

Table 11: StateLedger LoCoMo judge results by benchmark category.

| Category | Correct | Questions | Accuracy (%) |
|---|---|---|---|
| Single-hop (4) | 810 | 841 | 96.3 |
| Multi-hop (1) | 275 | 282 | 97.5 |
| Temporal (2) | 305 | 321 | 95.0 |
| Open-domain (3) | 86 | 96 | 89.6 |
| Non-adversarial (1–4) | 1,476 | 1,540 | 95.8 |
| Adversarial (5) | 446 | 446 | 100.0 |
| All categories | 1,922 | 1,986 | 96.8 |

**Published reference protocols.** The Mnemis rows in Table 3 are reported by Tang et al. [2026], including its GPT-4.1-mini setting and the $k = 30$ retrieval variant. Hindsight, Zep, Mem0-Graph, and Mem0 values are from Table 4 of Latimer et al. [2025]; method citations identify the original systems. The token-efficient Mem0 SDK row is the April 2026 result reported by Mem0 [2026]. MemMachine values are from its September 2025 report [MemMachine, 2025], using a GPT-4o-mini reader/judge, text-embedding-3-small, and an agent with multiple searches. TiMem values are from Table 1 of Li et al. [2026], using GPT-4o-mini-2024-07-18 for recall/generation, $k = 20$, and a mean over three runs. Source-specific judges, budgets, category aggregation, and averaging are retained.

The gpt-4o-mini reader receives ranked StateLedger evidence and returns concise answers; the judge receives the question, reference answer or rubric required by the protocol, and the model response. Judge outputs are stored as evaluation artifacts and are not fed back into memory.

Table 12: LoCoMo execution protocol.

| Stage | Archived setting |
|---|---|
| Prepare | convert sessions into ordered SourceEvents and content-addressed artifacts; apply the shared path/register/flush/checkpoint pipeline; preserve speaker and date metadata |
| Retrieval | path-addressed ranking over checkpoints, CommitPages, cue views, and artifacts with top_k=50; no benchmark-specific memory store |
| Generation | gpt-4o-mini, generation_top_k=10, max_context_chars=12000, max_tokens=64, temperature=0 |
| Judge | gpt-4o-mini, max_completion_tokens=48, temperature=0 |

Table 13: LoCoMo retrieval diagnostics by category.

| Cat. | N | Any@1 (%) | Any@10 (%) | Any@50 (%) | Full@10 (%) | Full@50 (%) | AnsStr@10 (%) | MRR | Sec. |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 282 | 38.7 | 77.7 | 92.6 | 22.3 | 46.8 | 15.6 | 0.508 | 1.034 |
| 2 | 321 | 64.8 | 92.8 | 96.9 | 88.2 | 94.4 | 25.2 | 0.750 | 1.091 |
| 3 | 96 | 25.0 | 53.1 | 74.0 | 32.3 | 51.0 | 9.4 | 0.343 | 1.269 |
| 4 | 841 | 51.1 | 86.4 | 94.2 | 85.6 | 93.7 | 56.8 | 0.638 | 1.243 |
| 5 | 446 | 19.3 | 45.3 | 69.7 | 44.4 | 69.3 | 0.0 | 0.284 | 1.212 |

### D.5 Evo-Memory Multi-turn details

Evo-Memory Multi-turn is evaluated in the environment loop. At each turn, the policy receives only the current observation, the allowed or admissible action interface when provided by the benchmark, and the current StateLedger task state. After execution, the observation, action outcome, and typed state delta are converted by the environment adapter into SourceEvents and processed by the same path/register/flush pipeline. Progress and success are computed after prediction for reporting and diagnostics; no expert plan, hidden simulator fact, success label, or score-derived progress label is used for action selection or ledger construction.

**Reference protocol.** The reference block of Table 4 reproduces the Claude 3.7 Sonnet results in Table 2 of Wei et al. [2025]. The two blocks use their respective controllers/backbones, action budgets, and task streams. Our adapter settings appear in Table 14.

Table 14: Evo-Memory Multi-turn execution protocol.

| Task family | Archived setting |
|---|---|
| ALFWorld | 134 AgentBoard eval_out_of_distribution tasks; success is raw ALFWorld/TextWorld info["won"]==True; timeout done=True is not success |
| BabyAI | seed=1234, seeds_per_level=4, 120-step execution cap; optional gpt-4o-mini controller with llm_interval=5 |
| PDDL | max_depth=80, max_expansions=40000 for the fixed symbolic controller; executed transitions, not search internals, become SourceEvents and are flushed through the same CommitPage protocol |
| ScienceWorld | test.jsonl task file, max_steps=30; typed facts, measurements, container state, and action outcomes become SourceEvents/artifact references under the same StateLedger pipeline |

Table 15: Evo-Memory complete final task-level table.

| Task | N | Success (%) | Progress (%) | Steps | Cues | Sec. |
|---|---|---|---|---|---|---|
| ALFWorld | 134 | 100.0 | 100.0 | 14.99 | 14.99 | 0.530 |
| BabyAI | 112 | 75.0 | 83.4 | 40.56 | 240.63 | 0.095 |
| PDDL | 60 | 100.0 | 100.0 | 20.38 | 203.83 | 0.316 |
| ScienceWorld | 90 | 73.3 | 90.5 | 14.60 | 313.83 | 1.344 |
| Overall avg. | 4 | 87.1 | 93.5 | – | – | – |
| Episode-weighted avg. | 396 | 86.9 | 93.1 | – | – | – |

Each executed transition is a hard-boundary SourceEvent, so commits equal steps.

Table 16: BabyAI results for all 28 levels. Each level has four episodes and 100% grounding accuracy; commits equal steps. Success and progress are percentages, and Sec. is time per episode.

| Level | Succ. | Prog. | Steps | Cues | Sec. | Level | Succ. | Prog. | Steps | Cues | Sec. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| level_01 | 100.0 | 100.0 | 6.50 | 32.50 | 0.026 | level_15 | 100.0 | 81.2 | 92 | 1,012 | 0.211 |
| level_02 | 100.0 | 100.0 | 8 | 35 | 0.018 | level_16 | 100.0 | 93.8 | 37.75 | 445 | 0.079 |
| level_03 | 100.0 | 100.0 | 3.75 | 18.75 | 0.004 | level_17 | 75.0 | 75.0 | 37.50 | 189 | 0.068 |
| level_04 | 100.0 | 100.0 | 3 | 15 | 0.002 | level_18 | 50.0 | 66.7 | 63.25 | 346.25 | 0.158 |
| level_05 | 100.0 | 100.0 | 4.50 | 22 | 0.015 | level_19 | 100.0 | 100.0 | 4 | 24 | 0.007 |
| level_06 | 50.0 | 56.2 | 61 | 305 | 0.071 | level_20 | 100.0 | 100.0 | 2 | 11.50 | 0.005 |
| level_07 | 50.0 | 68.3 | 61.75 | 277 | 0.133 | level_21 | 75.0 | 75.0 | 51.75 | 310.50 | 0.093 |
| level_08 | 100.0 | 100.0 | 6.50 | 47.50 | 0.013 | level_24 | 75.0 | 83.3 | 33.50 | 197.50 | 0.068 |
| level_09 | 100.0 | 100.0 | 3.25 | 16.25 | 0.011 | level_25 | 50.0 | 58.3 | 66.75 | 333.75 | 0.090 |
| level_10 | 100.0 | 100.0 | 3.50 | 15.75 | 0.007 | level_27 | 0.0 | 58.3 | 120 | 720 | 0.757 |
| level_11 | 100.0 | 100.0 | 2.75 | 13.25 | 0.010 | level_28 | 0.0 | 62.5 | 120 | 600 | 0.313 |
| level_12 | 75.0 | 83.3 | 49.50 | 277.50 | 0.079 | level_29 | 0.0 | 58.3 | 120 | 600 | 0.143 |
| level_13 | 100.0 | 100.0 | 2.25 | 13.50 | 0.002 | level_31 | 100.0 | 85.0 | 47.50 | 237.50 | 0.067 |
| level_14 | 100.0 | 100.0 | 3.50 | 21.75 | 0.009 | level_32 | 0.0 | 30.0 | 120 | 600 | 0.200 |

Table 17: PDDL complete domain-level results.

| Domain | N | Success (%) | Progress (%) | Grounding (%) | Steps | Commits | Cues | Sec. |
|---|---|---|---|---|---|---|---|---|
| barman | 20 | 100.0 | 100.0 | 100.0 | 14.45 | 14.45 | 144.50 | 0.067 |
| blockworld | 10 | 100.0 | 100.0 | 100.0 | 15 | 15 | 150 | 0.072 |
| gripper | 20 | 100.0 | 100.0 | 100.0 | 28.80 | 28.80 | 288 | 0.822 |
| tyreworld | 10 | 100.0 | 100.0 | 100.0 | 20.80 | 20.80 | 208 | 0.043 |
| ALL | 60 | 100.0 | 100.0 | 100.0 | 20.38 | 20.38 | 203.83 | 0.316 |

Table 18: ALFWorld diagnostics by difficulty and operation. Success and progress are 100% in every group; commits equal steps. JSON sizes are mean bytes per episode.

| Group | N | Steps | StateLedger JSON | Trajectory JSON |
|---|---|---|---|---|
| easy | 24 | 17.67 | 26546.6 | 3771.6 |
| hard | 110 | 14.41 | 23518.9 | 3150.7 |
| clean | 31 | 12.87 | 21705.4 | 2880.5 |
| cool | 21 | 12.19 | 23780.4 | 2835.4 |
| examine | 18 | 12.11 | 17620.8 | 2506.2 |
| heat | 23 | 10.61 | 20,729 | 2491.8 |
| place | 41 | 21.76 | 30683.0 | 4532.4 |

Table 19: ScienceWorld difficulty-level diagnostics.

| Difficulty | N | Success (%) | Progress (%) | Steps | Commits | Cues | Sec. |
|---|---|---|---|---|---|---|---|
| easy | 34 | 88.2 | 94.1 | 7.38 | 7.38 | 128.38 | 0.753 |
| hard | 56 | 64.3 | 88.2 | 18.98 | 18.98 | 426.43 | 1.703 |
| ALL | 90 | 73.3 | 90.5 | 14.60 | 14.60 | 313.83 | 1.344 |

### D.6 Scope of component diagnostics

Table 5 uses the 5 May 2026 component-diagnostic archive, pooling available records across evaluation rounds. The LongMemEval subset contains final-correct questions with complete Stage 3 retrieval, Stage 4 bundle, and Stage 5 answer records: 489 on S and 385 on M, including 90 multi-session M questions. Field removal measures pre-generation answer-token coverage in this subset, quantifying each component's evidence support. The LoCoMo retrieval-budget diagnostic covers all 1,986 questions and measures full-evidence recall after retrieval.

### D.7 Reader portability and drift sensitivity

**Reader portability.** We froze retrieval outputs for 50 LongMemEval-S and 50 LongMemEval-M questions and reran four memory methods with three alternative readers. Within each reader, all methods use identical question ids, prompt, decoding configuration, a 12k-character evidence cap, and a 48-token answer cap. Token-F1 is computed directly from generated answers without an LLM judge. We report paired question-level bootstrap intervals with 10,000 resamples. For GPT-5.6-Sol, we used provider-default decoding and averaged token-F1 per question over two complete runs before bootstrapping.

Table 20: Matched token-F1 (%) across alternative readers on 100 fixed questions.

| Reader | StateLedger | Hybrid | Session BM25 | Rolling summary | Smallest paired gain (pp; 95% CI) |
|---|---|---|---|---|---|
| Qwen2.5-72B-Instruct | **36.71** | 17.53 | 17.41 | 16.78 | +19.18 [11.73, 26.99] |
| GPT-5.6-Sol (two runs) | **34.81** | 14.51 | 15.96 | 12.65 | +18.85 [11.40, 26.39] |
| Qwen3-Next-80B-A3B-Instruct | **39.40** | 22.18 | 21.37 | 18.62 | +17.22 [9.67, 25.05] |

*Note.* Hybrid denotes BM25+MiniLM session retrieval. StateLedger ranks first in all nine reader–control comparisons; all nine paired 95% intervals exclude zero. The same ordering holds separately on S and M.

**Drift sensitivity.** We sweep the drift threshold over 0.10, 0.20, and 0.35 across five replay seeds. Table 21 reports events per CommitPage, durable commit ratio, and durable current-state accuracy. Raising the threshold from 0.10 to 0.35 reduces the commit ratio from 37.88% to 17.14%, while all three settings retain 100% durable current-state accuracy.

Table 21: Drift-threshold sensitivity over five replay seeds.

| Drift threshold | Events/Commit | Durable commit ratio | Durable current-state accuracy |
|---|---|---|---|
| 0.10 | 2.64 | 37.88% | **100.0%** |
| 0.20 | 4.17 | 23.97% | **100.0%** |
| 0.35 | 5.83 | 17.14% | **100.0%** |

### D.8 Asset licenses and code availability

The benchmarks and model families are credited in the main text. Table 22 gives the upstream licenses for the data, environment code, and open model assets used in the evaluation. Environment-code licenses and dataset licenses are listed separately. API-based models are accessed under their providers' service terms. The flower frames in Figure 1 are credited to Ajith Samuel under CC BY 3.0 in the figure caption.

Table 22: Third-party assets and upstream licenses. Links identify the corresponding releases or license files.

| Asset | License and upstream source |
|---|---|
| LongMemEval | MIT (dataset; code). |
| LoCoMo | CC BY-NC 4.0 for the repository release. |
| AgentBoard | Apache-2.0 for code; GPL-2.0 for data, including the PDDL evaluation split. |
| ALFWorld environment | MIT for environment code. |
| BabyAI / MiniGrid | BSD-3-Clause for original BabyAI code; Apache-2.0 for MiniGrid environment code. |
| ScienceWorld environment | Apache-2.0 for environment code. |
| PDDLGym | MIT for the upstream environment code. |
| MiniLM sentence embeddings | Apache-2.0 for all-MiniLM-L6-v2. |
| Qwen2.5-72B-Instruct | Qwen License Agreement. |
| Qwen3-Next-80B-A3B-Instruct | Apache-2.0. |

Public release of the StateLedger implementation and reproduction scripts is planned after the conference. The experiments use existing datasets and model assets credited in Table 22.

## 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] The abstract and introduction describe the path-addressed memory substrate, structural guarantees, and empirical results. Sections 4 and 6 specify the theoretical and evaluation scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes] Section 6 discusses the write/read interface, query scope, workload characteristics, and deployment evaluation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: [Yes] Section 4 states the main theoretical claims and Appendix C gives the formal model, assumptions, theorem statements, and complete proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: [Yes] Section 3 and Appendices A and D describe the method and benchmark protocols for reproducing the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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] The benchmark datasets are public and cited. Public release of the implementation and reproduction scripts is planned after the conference.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: [Yes] Appendix D reports datasets, splits, budgets, model settings, and evaluation protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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] Appendix D.2 reports fixed-test-set counts and Wilson 95% confidence intervals for the main binary accuracy/success metrics. Non-binary progress and storage diagnostics are reported as deterministic aggregates over the fixed benchmark instances.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: [Yes] Appendix D.2 reports hardware, memory, API use, and elapsed time; benchmark tables provide local timings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] The work uses public benchmark data and does not introduce new human-subject data collection, hidden user profiling, or deployment on real users. The experiments exclude gold answers and hidden simulator state from construction, retrieval, planning, and reader prompts.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes] The paper motivates auditable and efficient long-term agent memory. Section 6 identifies privacy enforcement, adversarial-write robustness, and multi-writer consistency as deployment considerations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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] The paper does not release a pretrained generative model, image generator, or scraped dataset with high misuse risk. The planned code release will contain an implementation and reproduction scripts, not a new high-risk model or private dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: [Yes] Appendix D.8 lists the benchmark, environment-code, and open-model licenses with upstream links. The original works are credited, API models use their providers' service terms, and the flower-video adaptation is attributed under CC BY 3.0.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A] This submission releases no new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: [N/A] The paper does not include crowdsourcing experiments or new research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: [N/A] The paper does not involve crowdsourcing or new human-subject experiments, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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] The paper describes where LLM readers, judges, and planners are used in the experimental protocols, including the model names, context limits, token limits, and temperature settings. No LLM weights are trained or modified by the proposed method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.