Beyond Semantic Organization: Memory as Execution State Management for Long-Horizon Agents

Anonymous ACL submission

Abstract

LLM-based agents increasingly tackle long-horizon tasks with interdependent decisions, where each action reshapes future constraints and intermediate errors can cascade. Existing RAG and agent memory systems organize histories by semantic similarity, retrieving content-relevant entries at decision time. We argue that this design mismatches execution-state dependencies: it fragments decision trajectories and mixes valid and erroneous traces, hindering coherent state reconstruction and error isolation. We propose MAGE (**M**emory as **A**gent-**G**uided **E**xploration), an active execution-state manager that stores interactions in a hierarchical state tree. The agent derives its state from the active root-to-current path, combining subgoal summaries, recent traces, and hints from prior branches. Four coupled operations maintain the tree: `Grow` records new traces, `Compress` summarizes completed subgoals, `Maintain` validates summaries, and `Revise` restores a target boundary and resumes on a new branch. This design bounds context growth while preserving state integrity and isolating flawed segments from the active path. Experiments on MemoryArena show that MAGE improves task success rate by 27.5–127.7% over baselines, while reducing token consumption by 55.1%.

1 Introduction

With the growing ability of large language models (LLMs) to interact with complex environments through tool use and multi-step reasoning, LLM-based agents are increasingly deployed for long-horizon tasks with interdependent decisions (Yao et al., 2022; Xie et al., 2024; Zhou et al., 2024; Lobo et al., 2025; He et al., 2025). These tasks involve hundreds of steps where each action reshapes future choices, and intermediate errors can cascade to invalidate subsequent progress. Unlike recall-oriented memory benchmarks that answer questions over past conversations or agentic traces (Maharana et al., 2024; Wu et al., 2025; Zhao et al., 2026), the interdependent long-horizon agent tasks we study require maintaining a coherent, evolving execution state, as each decision depends on the cumulative outcome of prior steps.

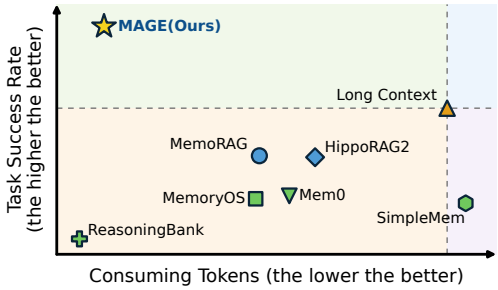

Figure 1: **Paradigm comparison on long-horizon agent tasks** (MemoryArena). Long-context approach achieves strong task performance but with high token cost, whereas baselines reduce context at the risk of losing state dependencies and underperforming. By managing memory as an execution-state tree, MAGE reaches the ideal upper-left region with the highest task performance and fewer tokens than long-context.

This requirement becomes harder as exploration history grows beyond the model's effective context window (Packer et al., 2023; Liu et al., 2024). To address this, recent works introduce memory systems (Chhikara et al., 2025; Xu et al., 2025; Rasmussen et al., 2025; Kang et al., 2025; Hu et al., 2025; Zhang et al., 2025; Liu et al., 2026) that record past information as compact entries and retrieve relevant ones on demand. Yet recent benchmarks reveal a counter-intuitive pattern: these systems often fail to improve long-horizon agent performance and sometimes underperform approaches that simply retain the full history in context (He et al., 2026; Zhao et al., 2026). As shown in Figure 1, many such systems consume substantial tokens while still trailing the long-context approach.

We argue that a key cause lies in the shared design philosophy. Although these systems vary in their data structures, ranging from flat vector stores

to entity-relation graphs to hierarchical architectures, they generally rely on *semantic relationships* to organize and retrieve information, surfacing entries by their content relevance to the current query rather than their role in the execution trajectory.

Such similarity-driven organization leads to two recurring problems when handling interdependent long-horizon tasks. First, it causes **state fragmentation** that weakens execution state integrity. The agent's execution state is built up through a chain of dependent decisions where each step is conditioned on the context established by prior steps. Existing systems, even those with graph structures, organize this state as entries linked by semantic or topical relationships rather than state dependencies, discarding critical execution context that binds them together. As a result, the system may fail to reconstruct a complete, coherent execution state, leading to erroneous actions based on incomplete information (Figure 2(a)–(b)).

Second, it hinders **effective error isolation**. Similarity-based memory mixes entries from different trajectories or exploration attempts in the same relevance space, so erroneous and valid traces can be surfaced together and contaminate subsequent reasoning (Figure 2(d)). Without explicit path structure and revision boundaries, it is also difficult to trace an error back to its origin or isolate the affected segment, allowing errors to propagate and accumulate over the course of execution.

These observations suggest that memory for interdependent long-horizon agents should shift from a similarity-driven archive to an *execution-state manager*. To this end, we propose MAGE (**M**emory as **A**gent-**G**uided **E**xploration), which treats memory as an execution state structure rather than a pool of retrievable facts. MAGE organizes the agent's history as a persistent two-layer hierarchical state tree. The bottom layer records the step-by-step action-observation trace, while the top layer stores summaries generated at subgoal or decision boundaries. This boundary-aware compression reduces context without interrupting an active trace or breaking execution-state integrity. The current execution state is read from the active tree path instead of being assembled from semantically similar entries, combining compressed state, recent raw state, and execution hints from sibling branches. This path-based representation addresses state fragmentation by keeping the agent-facing state coherent while still bounding the context size.

Building on this tree, MAGE further supports error isolation by making memory an agent-manipulable object rather than a shared pool of mixed entries. Through a closed-loop execution cycle, `Grow` extends the raw trace and `Compress` summarizes the accumulated trace at subgoal or decision boundaries. Before a new summary becomes trusted memory, `Maintain` validates the summary and its underlying trace against the task, catching missing information or execution errors before they propagate. If an error is detected, `Revise` restores the execution state to the target boundary and resumes execution as a new branch. The erroneous segment is therefore excluded from the active path, while the valid progress before the target boundary is preserved, isolating the error from subsequent decisions. As shown in Figure 1, MAGE occupies the optimal upper-left quadrant, achieving stronger task progress with lower token consumption.

Our contributions are as follows. (1) We propose MAGE, which organizes agentic memory as a two-layer hierarchical tree whose root-to-current path provides a complete execution state by construction, shifting memory from similarity-driven retrieval to compact execution-state management. (2) We design four coupled operations that make this tree an agent-manipulable object, forming a closed-loop state-management cycle that isolates errors into separate branches and keeps the active execution state free from erroneous traces. (3) Evaluate on MemoryArena, MAGE improves the task success rate by 27.5–127.7% over baselines on average, while reducing token consumption by 55.1% compared with the long-context approach.

2 Background and Related Work

2.1 Problem Setting

Long-horizon agent tasks with interdependent decisions can be formulated as a Markov decision process (MDP) (Bellman, 1957; Yao et al., 2023). At step t, the environment state $s_t \in \mathcal{S}$ evolves deterministically as $s_{t+1} = T(s_t, a_t)$ after action $a_t \in \mathcal{A}$, and the agent receives an observation o_t describing the resulting state. As a result, the interaction history is $h_t = (a_1, o_1, \ldots, a_t, o_t)$; as t grows, this history can exceed the model's effective context window (Packer et al., 2023; Park et al., 2023; Liu et al., 2024; Shinn et al., 2023), making it the central challenge to organize h_t compactly while still supporting complete state reconstruction.

This formulation highlights two requirements. First, since s_t is determined by previous actions

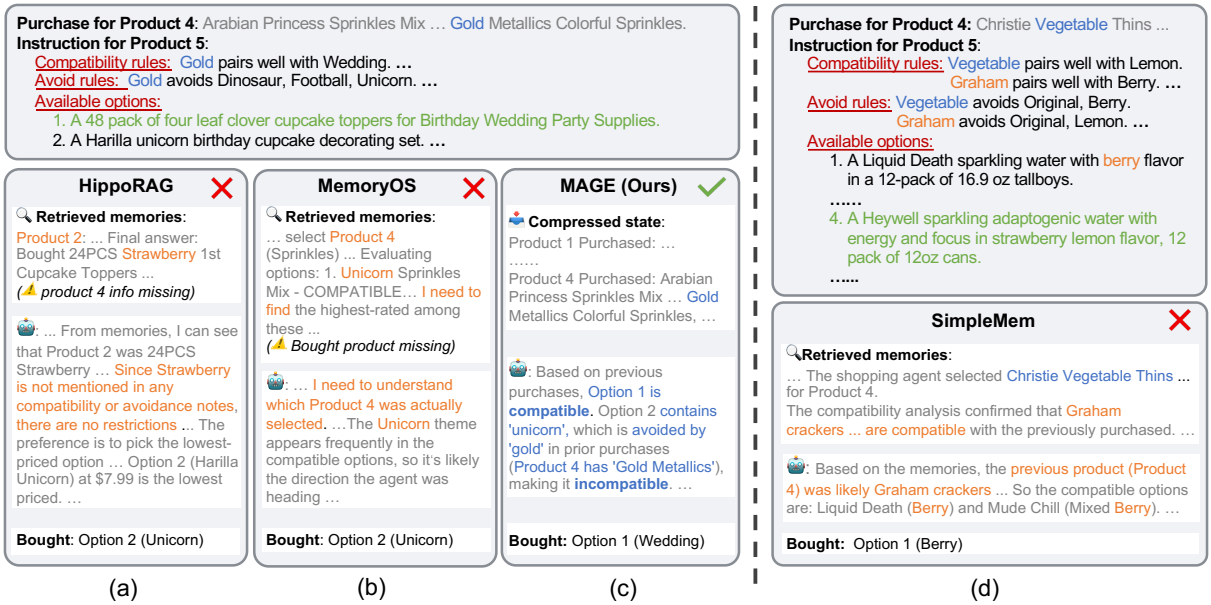

Figure 2: **Case study of baseline failures in MemoryArena shopping tasks**. In bundled shopping, each purchase must satisfy constraints induced by previously bought products. Cases (a)–(c) show state fragmentation: HippoRAG retrieves only Product 2 information, while MemoryOS retrieves Product 4 exploration traces but not the final purchased item; both miss that Product 4 contains *gold* and choose an incompatible *unicorn*-themed option. In contrast, MAGE preserves the execution state and selects the compatible *wedding* option. Case (d) shows error contamination, where SimpleMem retrieves mixed evidence from correct and incorrect trajectories and buys a *berry*-flavored item that violates the current *vegetable*-related avoidance rule.

$(a_1, \ldots, a_t)$, a sufficient memory representation must preserve the decision chain on which each step depends rather than only relevant entries. Second, if an action a_k is erroneous, downstream states $s_{k+1}, \ldots, s_t$ may become invalid; recovery therefore requires identifying the error origin, reverting to s_k, and re-executing from that point. This distinguishes our setting from traditional memory benchmarks (Maharana et al., 2024; Wu et al., 2025; Zhao et al., 2026; Jung et al., 2026), which mainly test recall of facts, preferences, or events from past conversations or traces. Since answers in these benchmarks do not alter the environment or invalidate future states, they measure retrieval fidelity rather than dynamic execution-state management.

2.2 Memory and Retrieval Systems for Agents

A natural approach to managing long histories is retrieval-augmented generation (RAG), which augments the LLM context with information retrieved from an external store. Existing RAG methods include *direct retrieval* with sparse or dense matching (Robertson and Zaragoza, 2009; Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020), *iterative retrieval* with query refinement (Borgeaud et al., 2022; Ma et al., 2023), *graph-structured RAG* for multi-hop reasoning (Edge et al., 2024; Gutierrez et al., 2024; Gutiérrez et al., 2025), and *memory-augmented RAG* that uses a lightweight model to form global memory or retrieval clues (Qian et al., 2025). These methods are effective for grounding generation in external knowledge, but the retrieved corpus is typically static and independent of the agent's action-conditioned state.

Agent memory systems instead store the agent's evolving history. They differ in storage design: *flat* systems keep independent records retrieved by embedding similarity (Chhikara et al., 2025; Liu et al., 2026; Xu et al., 2025; Nan et al., 2025); *graph-based* systems organize memories through entity or event relations (Rasmussen et al., 2025; Chen et al., 2025a; Ji et al., 2026; Hu et al., 2026b); *hierarchical* systems maintain multiple granularities to balance detail and compression (Packer et al., 2023; Kang et al., 2025; Hu et al., 2026a; Zhang et al., 2026; Li et al., 2026); and *hybrid* systems combine granularities or narrative structures for compact coverage (Ye et al., 2026; Patel and Patel, 2025; Wang et al., 2025; Zhou et al., 2026). Other work improves retrieval with prospective indexing or retrospective reflection (Tan et al., 2025; Latimer et al., 2025; Logan, 2026).

Despite this diversity, these systems commonly

expose memory through similarity-driven update and retrieval: they maintain a store $\mathcal{M}$ via $\mathcal{M} \leftarrow \texttt{Update}(\mathcal{M}, a_t, o_t)$ and retrieve entries $\texttt{Retrieve}(\mathcal{M}, q) \rightarrow \{e_1, \ldots, e_k\}$ by semantic relevance to query q. This design reduces context length but does not preserve the path structure needed for long-horizon tasks with interdependent decisions. It therefore causes **state fragmentation**, where the decision chain defining s_t is scattered across semantic fragments, and insufficient **error isolation**, where valid and erroneous trajectories coexist in the same memory pool without structural boundaries for rollback. Figure 2 illustrates both failures on shopping tasks (He et al., 2026).

3 Method

To address the issues inherent in similarity-driven memory systems, we propose MAGE, which shifts agentic memory from passive semantic storage and retrieval to active execution state management. Figure 3 illustrates the overall design.

3.1 Overview

Cognitive science suggests that humans performing complex sequential tasks rely on coordinated neural mechanisms. Prefrontal cortex organizes behavior into hierarchical subgoals and chunks completed segments to free working memory for subsequent planning (Botvinick et al., 2009). Anterior cingulate cortex monitors execution and signals failures at subgoal boundaries before they propagate to downstream decisions (Yeung et al., 2004). After detecting errors, executive control selectively backtracks to the relevant boundary and repairs the affected segment while preserving unaffected goal structure (Duncan, 2001). This cycle of chunking, monitoring, and correction motivates a memory system that manages execution state actively rather than merely storing past information.

Motivated by this architecture, we propose MAGE (**M**emory as **A**gent-**G**uided **E**xploration), which represents the agent's execution history as a two-layer hierarchical state tree. The bottom layer records raw action-observation nodes in execution order, preserving fine-grained state dependencies. The top layer stores summary nodes that cover completed bottom-layer segments, progressively chunking long local traces into compact subgoal-level states. Together, these two layers keep the root-to-current path complete rather than fragmented, while bounding the context and retaining the boundaries needed for future revision. Based on this tree, MAGE constructs the agent-facing execution state $\mathcal{S}$, consisting of compressed summaries $\mathcal{C}$, recent raw trace $\mathcal{R}$, and execution hints $\mathcal{H}$ from previously explored branches and diagnostic notes.

Table 1: MAGE operations parallel the cognitive mechanisms underlying human complex task execution.

Cognitive Mechanism	Function	Operation
Hierarchical chunking (Botvinick et al., 2009)	Organize subgoals; free working memory	`Grow` + `Compress`
Error monitoring (Yeung et al., 2004)	Detect errors at subgoal boundaries	`Maintain`
Selective correction (Duncan, 2001)	Backtrack to error origin; correct affected branch	`Revise`

Given this state representation, we design four operations to maintain the tree and refresh $\mathcal{S}$ in a closed loop, mirroring the cognitive cycle summarized in Table 1. During execution, `Grow` appends each new action-observation pair to the bottom-layer tree, extending the recent raw trace $\mathcal{R}$. `Compress` moves completed raw segments from $\mathcal{R}$ into top-layer summaries in $\mathcal{C}$, freeing context while preserving subgoal boundaries. `Maintain` acts as a boundary-level error monitor, validating each new summary before it becomes trusted memory and recording diagnostic notes. Upon detecting an error, `Revise` provides selective correction by restoring $\mathcal{C}$ and $\mathcal{R}$ to the relevant boundary, injecting diagnostic feedback into $\mathcal{H}$, and resuming execution as a new branch from that point.

This closed-loop design directly addresses state fragmentation and ineffective error isolation. First, MAGE constructs the current execution state from one path of the tree, combining compressed summaries for completed segments with raw traces for recent steps instead of retrieving disconnected memory entries based on similarity. Second, boundary-level maintenance and revision prevent erroneous segments from entering or contaminating the active execution state. When an error is detected, `Revise` branches from the target boundary, isolating the affected segment while preserving valid progress elsewhere. We next present the hierarchical execution state tree and the four state-transition operations.

3.2 Hierarchical Execution State Tree

MAGE organizes the agent's execution history as a two-layer hierarchical tree with a unified node structure (Table 2). The bottom layer records every

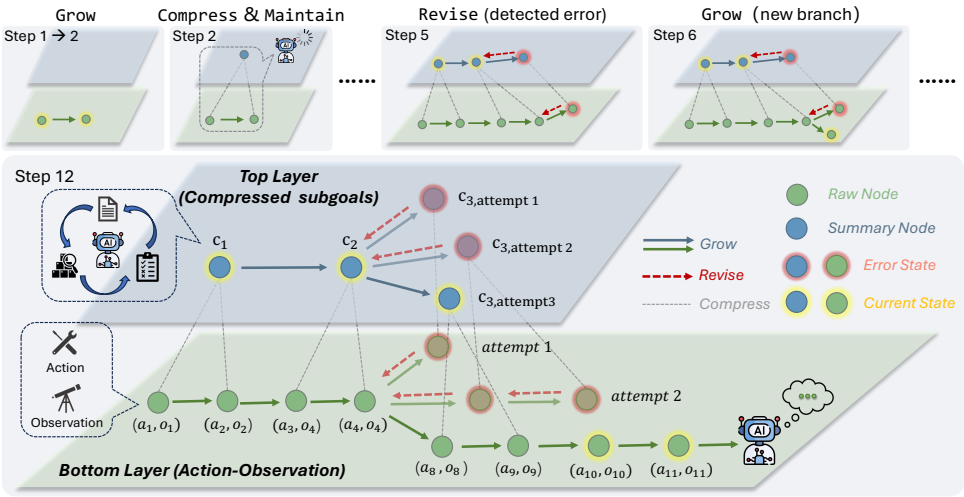

Figure 3: **Overview of MAGE.** MAGE maintains a two-layer execution-state tree: raw action-observation nodes grow in the bottom layer, while completed subgoals are compressed to the top layer. When an error is detected, `Revise` restores the target boundary and resumes exploration along a new branch, preserving unaffected progress.

Table 2: Node structure of the execution state tree.

Field	Description
`id`	Unique identifier
`content`	Action-observation pair (bottom layer) or compressed summary (top layer)
`parent`	Pointer to parent node
`children`	Set of child node pointers
`cover_nodes`	Ordered bottom-node pointers covered by this summary (top layer only)
`note`	Diagnostic feedback (top layer only)

raw action-observation pair as a node, preserving the fine-grained state of the execution. The root-to-current path through this layer yields the complete execution trajectory, while children of each node expose previously explored alternatives that can help the agent avoid repeating errors. When the agent revises a decision, new actions branch as siblings of the failed path, structurally isolating erroneous traces from valid ones. The top layer compresses contiguous bottom-layer segments into summary nodes, bounding the context as the task progresses. Each top-layer node corresponds to a completed subgoal, and we apply `Maintain` and `Revise` at these subgoal boundaries, the same locus where the prefrontal cortex chunks completed segments and the anterior cingulate cortex monitors for errors before they propagate (Table 1).

At runtime, MAGE navigates this two-layer structure with pointers p_b and p_t, tracking the agent's current positions in the bottom and top layers, respectively, and a global step ℓ assigning monotonically increasing ids to newly created nodes.

Based on the hierarchical tree, MAGE derives the agent's execution state $\mathcal{S}$ by composing three parts: (1) *compressed state* $\mathcal{C}$, consisting of top-layer summaries along the root-to-p_t path each annotated by its step id, allowing the agent to revise failed subgoals from the corresponding boundary upon detecting errors; (2) *raw state* $\mathcal{R}$, the bottom-layer nodes accumulated since the last compression that provide fine-grained recent context; and (3) *execution hint* $\mathcal{H}$, which surfaces children of p_b and p_t to reveal previously explored alternatives, along with diagnostic feedback from prior failed attempts. This representation equips the agent with a complete execution state and corrective guidance, sustaining coherent decision-making over long-horizon tasks with interdependent steps.

3.3 State-Transition Operations

The hierarchical tree becomes an active execution manager through four operations that transition the execution state as the agent progresses. Algorithm 1 in Appendix provides the pseudocode.

Grow. When the agent executes an action and receives an observation, MAGE automatically invokes `Grow` to update the bottom layer. If p_b already has a child with identical content from a prior exploration, the pointer advances to that child, merging back into the explored path without duplication; otherwise, a new node is created and linked as a child of p_b:

$$p_b' = \begin{cases} c, & \exists c \in p_b.\text{children} : c.\text{content} = (a, o), \\ \text{NewNode}(\ell, (a, o)), \ \ell \leftarrow \ell + 1, & \text{otherwise.} \end{cases}$$

The raw state $\mathcal{R}$ is then extended with the new action-observation pair, and the execution hint $\mathcal{H}$ is updated with children of the current node:

$$\mathcal{R}' = \mathcal{R} \| (a, o), \quad \mathcal{H}' = \text{Update}(\mathcal{H}, p_b'.\text{children})$$

This informs the agent of continuations attempted in prior explorations and helps it avoid repeating failed strategies.

Compress. `Compress` bounds context growth by replacing a completed bottom-layer segment with a top-layer summary node, freeing space while preserving the decision boundary needed for later recovery. It is invoked when the agent marks a subgoal complete with summary content provided as an argument, or by MAGE as a fallback when the raw state $\mathcal{R}$ exceeds a length threshold. Through this boundary-aware compression, MAGE avoids interrupting unfinished subgoals and keeps the state compact without discarding dependencies needed by future decisions.

Operationally, `Compress` traces the bottom-layer tree from p_b back to the last compressed boundary (recorded as p_t.cover_nodes$[-1]$), and uses the traversed nodes in execution order as the new summary node's cover_nodes. If a child of p_t already covers the same bottom nodes, it is reused; otherwise, a new summary node is created and inserted into the top-layer tree:

$$\begin{aligned} C_b &= \text{Trace}(p_t.\text{cover_nodes}[-1], p_b), \\ \ell_b &= p_t.\text{cover_nodes}[-1].\text{id}, \\ p_t' &= \begin{cases} c, & \exists c \in p_t.\text{children} : c.\text{cover_nodes} = C_b, \\ \text{NewNode}(\ell_b, \text{sum_content}, C_b), & \text{otherwise.} \end{cases} \end{aligned}$$

Then, `Compress` clears the current raw state $\mathcal{R}$, appends the summary content to the compressed state $\mathcal{C}$, and updates the execution hint $\mathcal{H}$ with children of p_t':

$$\mathcal{R}' = \emptyset, \quad \mathcal{C}' = \mathcal{C} \| (p_t'.\text{content}), \quad \mathcal{H}' = \text{Update}(\mathcal{H}, p_t'.\text{children}).$$

This exposes previously attempted subgoals from the new boundary while keeping the compressed state compact.

Maintain. Immediately after compression, `Maintain` validates the just-completed subgoal before the new summary becomes a trusted part of memory. This check protects the execution state from incorrect memory writes, allowing MAGE to detect missing information, unsatisfied task requirements, or broken dependencies before such errors accumulate. An LLM examines the compressed subtree together with the summary content and task instruction:

$$f = \text{LLM}(\text{task_inst}, \ p_t.\text{cover_nodes}, \ p_t.\text{content}).$$

If validation passes, execution continues. Otherwise, `Maintain` records the diagnostic feedback f in p_t.note and returns a failure signal with the revision target p_t.id.

Revise. Triggered by a `Maintain` failure or invoked proactively by the agent upon detecting an error, `Revise` restores the active path to the target step ℓ_t, which is either returned by `Maintain` or selected from exposed compressed-state boundaries. MAGE rolls both pointers backward until the target is reached, where re-exploration begins:

$$(p_t', p_b') = \text{Restore}(p_t, p_b, \ell_t).$$

The compressed state $\mathcal{C}$ and raw state $\mathcal{R}$ are reverted to these positions, while the execution hint $\mathcal{H}$ is updated with diagnostic feedback and alternatives from the restored nodes, providing extra guidance that helps avoid repeating the same error:

$$\begin{aligned} &\mathcal{C}' = \text{RestoreState}(p_t'), \quad \mathcal{R}' = \emptyset, \\ &\mathcal{H}' = \text{Update}(\mathcal{H}, p_t'.\text{children} \cup p_b'.\text{children} \cup \{f\}). \end{aligned}$$

Subsequent actions branch from this restored point as sibling paths, achieving error isolation without discarding valid progress on other branches.

4 Experiments

4.1 Experimental Setup

Benchmark. We evaluate on MemoryArena (He et al., 2026), an interdependent long-horizon benchmark where agents operate in a continuous Memory-Agent-Environment loop for up to hundreds of steps. Unlike conventional benchmarks that test static fact retrieval or question answering over past dialogues and traces (Maharana et al., 2024; Wu et al., 2025; Zhao et al., 2026), MemoryArena follows action-conditioned MDPs: each action can reshape future constraints, so success requires tracking the evolving execution state rather than recalling facts.

MemoryArena spans four domains with long dependency structures: *Bundled Web Shopping* (Yao et al., 2022), where the agent purchases a bundle of related products and later choices depend on earlier items; *Group Travel Planning* (Xie et al.,

2024), in which the agent coordinates multi-person itineraries to satisfy interdependent preferences; *Progressive Web Search* (Chen et al., 2025b), where the model answers complex queries progressively using information gathered from previous sub-queries; and *Formal Reasoning*, in which the agent proves complex claims through sequential derivations that build on previously established results.

Baselines. We compare MAGE against representative methods across three paradigms. *Long Context* retains full interaction history. *RAG systems* include HippoRAG2 (Gutiérrez et al., 2025), which builds a knowledge graph and applies Personalized PageRank for multi-hop retrieval, and MemoRAG (Qian et al., 2025), which uses a lightweight memory model to generate retrieval clues. *Memory systems* include Mem0 (Chhikara et al., 2025), which extracts and consolidates facts into graph-based memory, ReasoningBank (Ouyang et al., 2025), which distills reusable reasoning strategies from past experiences, MemoryOS (Kang et al., 2025), which maintains hierarchical storage layers with dynamic cross-level updating, and SimpleMem (Liu et al., 2026), which performs semantic compression and recursive consolidation for efficient memory management. All methods use the default hyperparameters from their original papers.

Model. All methods use Qwen3.6-27B (Qwen, 2026a) as the backbone LLM with ReAct (Yao et al., 2023) for agent exploration. Baselines requiring embeddings use Qwen3-8B-Embedding (Qwen, 2025). Inference runs on NVIDIA A100 GPUs (NVIDIA, 2020) with vLLM (Kwon et al., 2023) 0.20.0 under Python 3.12. Results under more backend models are reported in Appendix C.

Metrics. Following MemoryArena (He et al., 2026), we report average Task Success Rate (SR, %), Task Progress Score (PS, %), and total token consumption. SR measures full task completion: all subtasks must be correct in Shopping and Travel Planning, while the final subtask determines success in the other two domains. PS measures completed-subtask fraction, and token consumption includes prompt and generation tokens.

4.2 Main Results

Table 3 presents task performance across four domains. On tasks with complex state dependencies, RAG and memory-based baselines underperform the long-context approach, with relative SR drops of 38.0–66.0% on Web Shopping and 13.1–37.4% on Web Search. This gap suggests that similarity-driven retrieval fragments the execution state into isolated entries, discarding structural dependencies retained in full interaction history. The partial exceptions occur when dependencies can be recovered as sparse local evidence. In Formal Reasoning, baselines perform comparably or slightly better than the long-context approach because mathematical dependencies are explicit and sparse, enabling precise lemma retrieval. Similarly, HippoRAG2 achieves good results on Travel Planning because local constraints are anchored by stable entities and attributes (e.g., hotel–city, restaurant–cuisine), making graph retrieval effective for recovering candidate facts; yet full travel plans require cross-traveler constraints from prior decisions, and its fragmented retrieved triples do not preserve this evolving execution state, so its PS advantage does not translate into higher SR than MAGE.

Conversely, MAGE outperforms the long-context approach by average relative margins of 27.5% in SR and 19.3% in PS. This gain stems from treating memory as an active execution-state manager, not a passive archive. Modeling the trajectory as an active path within a two-layer hierarchical tree, MAGE maintains a coherent current state, reuses historical exploration to avoid recurring errors, and quarantines flawed segments into inactive branches to prevent contamination of the active path.

4.3 Token Efficiency

Regarding efficiency, Table 3 also shows token consumption per task. While RAG and memory baselines theoretically reduce context length, this benefit primarily materializes in document-heavy environments like Web Search, where they reduce token usage by 38.5–68.2% compared with the long-context approach. In domains with shorter observations and frequent state updates, the overhead of memory maintenance (e.g., extraction, query rewriting), compounded by the additional reasoning and execution steps required to synthesize fragmented retrieved states, frequently eclipses the compression savings. Consequently, systems like HippoRAG2 and Mem0 end up consuming 12.6–14.7% more tokens than the long-context approach on Web Shopping. In contrast, MAGE consistently reduces token usage by 32.9–71.4% across all domains. Unlike traditional memory systems that require continuous, token-heavy auxiliary LLM calls to extract entities or generate queries, MAGE

Table 3: Main results on MemoryArena. SR = Task Success Rate (%); PS = Task Progress Score (%); #tokens = average token consumption per task. Best results in **bold**, second best underlined. MAGE achieves the best task accuracy while reducing token consumption.

	Bundled Web Shopping			Group Travel Planning			Progressive Web Search			Formal Reasoning				
										Math		Physics		
Method	SR	PS	#tokens	SR	PS	#tokens	SR	PS	#tokens	SR	PS	SR	PS	#tokens
Long Context	0.3333	0.7578	*1528K*	0.0519	0.4070	*3211K*	0.4842	0.2803	*6045K*	0.4000	0.4124	**0.6500**	0.6977	*1782K*
HippoRAG2	0.2067	0.7189	*1720K*	0.0963	**0.5569**	*3153K*	0.3620	0.2133	*2865K*	**0.4500**	0.4153	0.6000	0.7093	*2268K*
MemoRAG	0.2067	0.7200	*1251K*	0.0481	0.4731	*2829K*	0.4208	0.2535	*3535K*	**0.4500**	0.4237	**0.6500**	**0.7209**	*1314K*
Mem0	0.1933	0.6822	*1753K*	0.0259	0.3498	*2786K*	0.3529	0.2206	*2715K*	0.4000	0.4040	0.5500	0.6977	*1853K*
ReasoningBank	0.1133	0.6033	*868K*	0.0000	0.2346	*1463K*	0.3032	0.1993	*1923K*	0.4250	0.4266	0.5500	0.6744	*1187K*
MemoryOS	0.2000	0.7044	*1448K*	0.0259	0.4405	*2984K*	0.3303	0.2084	*2973K*	**0.4500**	0.4209	0.5500	0.6977	*1452K*
SimpleMem	0.1600	0.6767	*2939K*	0.0222	0.3246	*4616K*	0.3575	0.2310	*3715K*	0.4000	0.4294	0.5500	0.6628	*1656K*
MAGE (Ours)	**0.3933**	**0.7778**	*1015K*	**0.1519**	0.5351	*1978K*	**0.5656**	**0.3790**	*1727K*	0.4250	**0.4492**	**0.6500**	0.6977	*1195K*

Table 4: Ablation study results. Each row removes one mechanism from the full MAGE.

Variant	Web Shopping			Travel Planning		
	SR	PS	#tokens	SR	PS	#tokens
MAGE (Full)	0.3933	0.7778	*1015K*	0.1519	0.5351	*1978K*
w/o Compress	0.3200	0.7233	*2469K*	0.0852	0.4496	*3539K*
w/o Maintain	0.3267	0.7389	*887K*	0.1000	0.4683	*1256K*
w/o Revise	0.3533	0.7378	*1157K*	0.1000	0.4708	*2551K*

maintains the tree structure deterministically and invokes auxiliary LLMs only during `Compress` and `Maintain` at natural subgoal boundaries. This design minimizes maintenance overhead without compromising state integrity.

4.4 Ablation Study

To validate the contribution of the main state-management mechanisms, we evaluate MAGE variants that remove one mechanism at a time. Table 4 shows that all three mechanisms contribute to reliable execution-state management.

Removing `Compress` consistently hurts task completion, reducing SR by 18.6% on Web Shopping and 43.9% on Travel Planning, while increasing token consumption to $2.4\times$ and $1.8\times$ that of the full model. This confirms that simply retaining the raw action-observation stream is not a sufficient substitute for memory: without boundary-aware compression, the active state becomes diluted by low-level traces, and `Maintain` must verify an increasingly long trajectory.

Removing `Maintain` leads to a different failure mode. Although it reduces token usage by avoiding boundary-level verification, SR drops by 16.9–34.2% on two domains. This result demonstrates that memory writes should be validated before they become trusted state. Otherwise, incomplete or erroneous subgoal summaries are committed to the active path and later decisions are conditioned on corrupted execution state.

Finally, removing `Revise` lowers SR by 10.2–34.2% on two domains. This shows that error detection alone is insufficient: after a failed boundary is identified, the agent must restore the corresponding state, branch away from the flawed segment, and continue from the preserved valid prefix; otherwise, the error remains on the active path and contaminates later decisions.

Notably, these weakened variants remain competitive with or superior to the baselines in Table 3. This suggests that organizing memory around the execution path mitigates the state fragmentation caused by semantic retrieval, while the full operation loop further prevents corrupted or failed segments from contaminating the active state.

5 Conclusion

We presented MAGE, a memory framework that reframes long-horizon agent memory as active execution-state management rather than similarity-driven retrieval. By organizing interaction history as a two-layer hierarchical state tree, MAGE preserves the active root-to-current execution path while compressing completed subgoals and exposing execution hints from previously explored branches. Its four coupled operations allow agents to extend traces, bound context growth, validate newly compressed states, and isolate erroneous segments through branching. Experiments on MemoryArena show that this design improves task success across diverse long-horizon domains while substantially reducing token consumption. These results suggest that preserving execution-state structure is a key principle for building reliable and efficient memory systems for real-world LLM agents.

Limitations

MAGE operates through prompting without task-specific training. Learning dedicated models for state compression and error detection could further improve the quality of execution-state management, which we leave for future work. Additionally, while we demonstrate the effectiveness of our approach on long-horizon tasks with interdependent decisions, extending MAGE to broader scenarios such as multimodal interaction or collaborative multi-agent workflows remains an important direction for future work.

References

Richard Bellman. 1957. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240.

Matthew M. Botvinick, Yael Niv, and Andrew G. Barto. 2009. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280.

Chunliang Chen, Ming Guan, Xiao Lin, Jiaxu Li, Luxi Lin, Qiyi Wang, Xiangyu Chen, Jixiang Luo, Changzhi Sun, Dell Zhang, and 1 others. 2025a. Telemem: Building long-term and multimodal memory for agentic ai. *arXiv preprint arXiv:2601.06037*.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, and 1 others. 2025b. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. In *European Conference on Artificial Intelligence*.

John S. Duncan. 2001. An adaptive coding model of neural function in prefrontal cortex. *Nature Reviews Neuroscience*, 2:820–829.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Junda He, Christoph Treude, and David Lo. 2025. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30.

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 J. McAuley, Yejin Choi, and Alex Pentland. 2026. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *CoRR*, abs/2602.16313.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. 2026a. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. *CoRR*, abs/2601.02163.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 32779–32798.

Yuyang Hu, Jiongnan Liu, Jiejun Tan, Yutao Zhu, and Zhicheng Dou. 2026b. Memory matters more: Event-centric memory as a logic map for agent searching and reasoning. *CoRR*, abs/2601.04726.

Shuo Ji, Yibo Li, and Bryan Hooi. 2026. MEMORY IS RECONSTRUCTED, NOT RETRIEVED: GRAPH MEMORY FOR LLM AGENTS. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*.

Seokwon Jung, Alexander Rubinstein, Arnas Uselis, Sangdoo Yun, and Seong Joon Oh. 2026. MEME: Multi-entity & evolving memory evaluation. *arXiv preprint arXiv:2605.12477*.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. 2025. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *CoRR*, 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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*.

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. 2026. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *CoRR*, abs/2601.02845.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for LLM agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Elita Lobo, Xu Chen, Jingjing Meng, Nan Xi, Yang Jiao, Yanhui Guo, Zhishen Huang, and Yan Gao. 2025. Hierarchical planning agent for web-browsing tasks. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Joe Logan. 2026. Continuum memory architectures for long-horizon LLM agents. *CoRR*, abs/2601.09913.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315. Association for Computational Linguistics.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 13851–13870.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *CoRR*, abs/2508.03341.

NVIDIA. 2020. Nvidia a100 tensor core gpu. https://www.nvidia.com/en-us/data-center/a100/. Accessed: 2025-04-01.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Daivik Patel and Shrenik Patel. 2025. ENGRAM: effective, lightweight memory orchestration for conversational agents. *CoRR*, abs/2511.12960.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference*, pages 2366–2377.

Qwen. 2025. Qwen3-embedding-8b. https://huggingface.co/Qwen/Qwen3-Embedding-8B. Accessed: 2026-05-01.

Qwen. 2026a. Qwen3.6-27b. https://huggingface.co/Qwen/Qwen3.6-27B. Accessed: 2026-05-01.

Qwen. 2026b. Qwen3.6-35b-a3b. https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Accessed: 2026-05-01.

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

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652.

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. 2025. 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*, pages 8416–8439.

Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2025. O-mem: Omni memory system for personalized, long horizon, self-evolving agents. *ArXiv*, abs/2511.13593.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, volume 2025, pages 86809–86836.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Juexiang Ye, Xue Li, Xinyu Yang, Chengkai Huang, Lanshun Nie, Lina Yao, and Dechen Zhan. 2026. Memweaver: Weaving hybrid memories for traceable long-horizon agentic reasoning. *CoRR*, abs/2601.18204.

Nick Yeung, Matthew Botvinick, and Jonathan Cohen. 2004. The neural basis of error detection: Conflict monitoring and the error-related negativity. *Psychological review*, 111:931–59.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025. G-memory: Tracing hierarchical memory for multi-agent systems. *CoRR*, abs/2506.07398.

Ningning Zhang, Xingxing Yang, Zhizhong Tan, and Weiping Deng. 2026. Himem: Hierarchical long-term memory for llm long-horizon agents. *arXiv preprint arXiv:2601.06377*.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, and 1 others. 2026. Ama-bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Yue Zhou, Xiaobo Guo, Belhassen Bayar, and Srinivasan H. Sengamedu. 2026. Amory: Building coherent narrative-driven agent memory through agentic reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3926–3938. Association for Computational Linguistics.

A Pseudocode of Four State-Transition Operations

Algorithm 1 provides the full pseudocode for the four state-transition operations described in Section 3.3.

B Experiment Details

B.1 Prompt Templates

We provide the prompt templates used in our experiments across the four MemoryArena domains (He et al., 2026). Each template specifies the task instruction and available action space that guide the agent's interaction with the environment.

Prompt for Bundled Web Shopping

You can interact with the webshop. Your goal is to purchase a bundle of items that are technically compatible and fit the budget. Each time you will receive one instruction with multiple options and you need to buy the best product among the options. For each option, search by the option text (do not miss any keywords) and browse multiple pages to find the most matched product, e.g., containing as many keywords mentioned in that option as possible, with the same weight, size, quantity, pack, brand, etc. Each option corresponds to only one best-matching product. The valid product may appear on later pages. Do NOT miss checking any option.

If given compatibility/avoid notes in the task, filter options/products by the compatibility notes (i.e., keywords of prior purchased products pair with keywords of the chosen product) and avoid notes (i.e., keywords of prior purchased products not pair with keywords of the chosen product). When applying compatibility/avoid notes, only consider exact keywords that appear in the text literally, do NOT infer the attributes of the product based on your internal knowledge. Compatibility and avoid notes should not be applied to the keywords of the chosen product itself, only apply them between prior purchased products and currently chosen product. If a product does not contain any keyword from the compatibility/avoid notes, it is compatible by default (no rule applies). Once you finish checking an option and find the corresponding product, store the information (product title, ASIN, price, rating if needed, etc) even the option is NOT compatible. Finally, compare all compatible products and navigate to the page of the product that satisfies the preference rule and buy it by clicking [Buy Now] button. You must buy exactly one product per task.

Constraint:

1. **Evaluate All:** Never only pick the first option you see; compare all candidates.
2. **Total Budget:** All items combined must not exceed $XX.
3. **Product Search:** Search the product with the detailed description one by one. For example, use "search[Product A]" but not "search[Product A, Product B, Product C]".
4. **Product Purchase:** You need to buy products on the order of the steps (i.e., Product 1 first, then Product 2, and so on).

Algorithm 1 State-Transition Operations of MAGE

Global Variable: Pointers of bottom-layer tree p_b and top-layer tree p_t, global step ℓ.
Execution State: $\mathcal{S} = (\mathcal{C}, \mathcal{R}, \mathcal{H})$ for compressed state, raw state, and execution hint.

```
function GROW(a, o)
    for all c ∈ p_b.children do
        if c.content = (a, o) then                ▷ merge node
            p_b ← c
            R ← R ‖ (a, o)
            H.UPDATE(p_b.children)
            return
        end if
    end for
    v ← NEWNODE(ℓ, (a, o));   ℓ ← ℓ + 1
    v.parent ← p_b;   p_b.children ← p_b.children ∪ {v}
    p_b ← v;   R ← R ‖ (a, o)
    H.UPDATE(p_b.children)
end function

function COMPRESS(m)                              ▷ input summary content
    b ← p_t.cover_nodes[−1]                       ▷ compressed boundary
    C ← TRACE(b, p_b)                             ▷ track nodes in execution order
    for all c ∈ p_t.children do                   ▷ merge node
        if c.cover_nodes = C then
            c.content ← m;   p_t ← c
            R ← ∅;   C ← C ‖ (p_t.content, p_t.id)
            H.UPDATE(p_t.children)
            return
        end if
    end for
    v ← NEWNODE(b.id, m, C)
    v.parent ← p_t;   p_t.children ← p_t.children ∪ {v}
    p_t ← v
    R ← ∅;   C ← C ‖ (p_t.content, p_t.id)
    H.UPDATE(p_t.children)
end function

function MAINTAIN(τ)                              ▷ input task instruction
    T ← FLATTEN(p_t.cover_nodes)                  ▷ flatten traces
    (q, f) ← LLM(τ, T, p_t.content)               ▷ LLM judge
    if q then
        return PASS
    else
        p_t.note ← f
        return (FAIL, f, p_t.id)
    end if
end function

function REVISE(f, b)                             ▷ input feedback and target step
    while p_t.id ≠ b do
        C.DELETE(p_t);   p_t ← p_t.parent
    end while
    C.DELETE(p_t)
    p_t ← p_t.parent                              ▷ skip the failed compressed node
    p_b ← p_t.cover_nodes[−1];   R ← ∅
    H ← p_t.children ∪ p_b.children ∪ {f}
end function
```

Prompt for Group Travel Planning

You are a travel planning agent that builds a multi-day itinerary for one traveler in a group by calling structured tools. Each task processes one traveler's query under shared group constraints; previously planned travelers are provided as context. You must search before selecting and only use option ids returned by search tools; never invent flights, restaurants, hotels, or attractions, and never construct option ids from names recalled in memory. Use the provided session id for every stateful action, and check cost and itinerary snapshots when budget or completeness is uncertain.

Planning strategy:
- For each day and slot (breakfast, lunch, dinner, attraction, accommodation), first search candidates, then select or book exactly one option from the returned ids.
- For relative constraints referencing another traveler, first recall the referenced slot from injected memory, convert to absolute search parameters, then search, verify each candidate against all conditions, and select the cheapest qualifying option.

Constraint:
1. **Grounded Selection:** Only select or book from ids returned by the most recent search; do not fabricate or reuse ids inferred from names in memory.
2. **Group Dependencies:** Use injected memory and previously planned travelers to derive absolute search constraints for the current traveler.
3. **Completion Criterion:** Call `Finish[done]` only when all required transportation legs are booked and every destination day has breakfast, lunch, dinner, attraction, and accommodation assigned.

Prompt for Progressive Web Search

You are a deep research agent solving a complex question decomposed into progressive sub-questions. All sub-questions constrain the same entity. Answer each question by interacting with a search engine, i.e., searching the corpus and reading documents; never Finish from internal knowledge or lock onto a candidate without corpus evidence. Keep your reasoning process compact and focused.

Search strategy:
- Use focused 3-6 word queries. Try different search angles with synonyms, related concepts, broader/narrower terms, or different clue combinations.
- Do not repeat search by only reordering keywords. If a search keyword pollutes results, drop it. If no relevant results are found, rethink entity type, wording, region/culture, or assumptions.
- Each search may return multiple relevant documents; read several promising ones, not just the first.
- Store brief notes for failed searches and each document you read, whether relevant or not.

Constraint:
1. You must search the corpus and read a document before answering. Never use internal knowledge as your answer. Find it in documents first.
2. Keep your reasoning process brief and output exactly one next action only; do not re-solve prior sub-questions, enumerate many candidates/search queries, or simulate observations.
3. Return the answer exactly as it appears in the document, using the complete, unabbreviated form (e.g., full legal name, full title, full official designation). Do not shorten or paraphrase.

Prompt for Formal Reasoning

You are a mathematical/physical reasoning assistant. Your task is to solve the math problem described in PROBLEM using the definitions and background provided in the context. All relevant definitions, lemmas, and background information are provided directly in the BACKGROUND section of the task. Refer to them as needed during your reasoning. Available tools: SymbolicReasoning (formal derivation), CodeExecutor (Python/SymPy computation), Compress (compress your derivation segment into a reusable summary), and Revise (undo to a previous step). You must perform at least one reasoning step (Think, SymbolicReasoning, or CodeExecutor) before calling Finish. Prefer SymbolicReasoning over Think for formal derivations.

WORKFLOW:
1. Use Think to analyse the problem and plan.
2. Use SymbolicReasoning to carry out formal derivation steps (preferred for proofs and formal arguments).
3. Use CodeExecutor to verify results with Python/SymPy.
4. Use Compress to compress your derivation into a summary.
5. Use Revise to undo to a previous step if needed.

Do NOT write markdown, headers, emojis, or any text outside the action call. Use LaTeX for mathematical expressions inside action parameters. Prefer SymbolicReasoning for formal derivations; Think is acceptable for brief analysis. At least one reasoning step before Finish. Use Compress to save intermediate results so that later sub-problems can reference them.

B.2 Baseline Evaluation Protocol

For all RAG and memory baselines, we follow the evaluation protocol of MemoryArena (He et al., 2026). Each method's storage $\mathcal{M}$ is initialized as empty at the start of each task. After each subtask finishes, the full interaction trace is written to $\mathcal{M}$ through the method's `Update` function. Before each action, we retrieve relevant entries from $\mathcal{M}$ using the current subtask trace as query, i.e., previous actions and observations, and append the retrieved content to the agent context for action generation.

B.3 Dataset Statistics

Table 5 summarizes the statistics of the four MemoryArena (He et al., 2026) domains used in our evaluation. We report the number of tasks, the average trace length, and the average number of execution steps for each domain. Trace length denotes the token length of the full-task interaction trajectory, including actions and observations, while execution steps count the number of agent-environment interaction steps in this trajectory. Both statistics are measured from long-context rollouts. The results show that the tasks require agents to operate over long traces with many steps, making sustained state

management essential for reliable performance.

Table 5: Dataset statistics for MemoryArena.

Domain	#tasks	Avg. Trace Len.	Avg. Steps
Web Shopping	150	24.53K	98.56
Travel Planning	270	25.19K	237.36
Web Search	221	101.92K	81.98
Math	40	21.34K	24.65
Physics	20	13.45K	13.75

C Results on Other Backend Models

To examine whether the benefits of MAGE generalize across backend models, we evaluate Qwen3.6-35B-A3B (Qwen, 2026b) on the representative Bundled Web Shopping domain from MemoryArena (He et al., 2026). As shown in Table 6, MAGE remains effective with this weaker model, improving SR by 35.3% over the long-context approach and by 77.0% and 155.6% over MemoRAG and MemoryOS, respectively. It also reduces token consumption by 33.2% relative to long-context, indicating that execution-state management is not tied to a particular backend model and that MAGE generalizes well.

Table 6: MemoryArena Results on Qwen3.6-35B-A3B.

Method	Metrics		
	SR	PS	#tokens
Long Context	0.2267	0.6978	4092K
MemoRAG	0.1733	0.6622	3888K
MemoryOS	0.1200	0.6144	3835K
MAGE	**0.3067**	**0.7256**	**2732K**

D Bounded Context Growth

Figure 4 shows that the long-context approach grows approximately linearly with execution steps because it continuously appends the action-observation history at every step. In contrast, MAGE bounds context growth through boundary-aware `Compress`, which chunks completed bottom-layer segments (subgoals) into compact top-layer subgoal summaries while retaining only the recent unfinished trace. This reduction does not break execution-state integrity: as shown in Table 3, MAGE reduces token consumption while improving the task performance compared with the long-context approach, indicating that the boundary-aware compressed state still preserves the dependencies needed for downstream decisions.

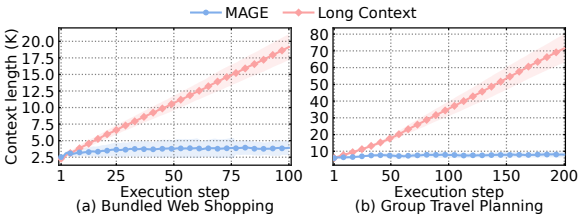

Figure 4: **Context Length vs. Execution Steps** (Average across tasks). MAGE bounds context growth through boundary-aware compression, whereas the long-context approach grows approximately linearly as execution steps accumulate.

E Case Studies

In this section, we present several case studies to illustrate the effectiveness of MAGE in managing long-horizon tasks with interdependent steps, comparing it with baseline methods.

Case Study A: SimpleMem Retrieves Fragmented Memories Instead of Current State

Task. `task_24_subtask_3`, Buy Product 4:

Compatibility notes:
- Fudge pairs well with Chocolate.
- Red pairs well with Gold.
- Pink pairs well with Pearl.
- Blue pairs well with Silver.
- Yellow pairs well with Rainbow.

Avoid notes:
- Fudge avoids Rainbow, Confetti, Pearl, Silver.
- Red avoids Green, Silver, Rainbow.
- Pink avoids Chocolate, Green, Orange.
- Blue avoids Gold, Red, Orange.
- Yellow avoids Silver, Gold.

Available options:
- A Gold Heart Sprinkles by Edible Party Supplies with 8 oz for baking and decorating cupcakes, cakes, cookies, and ice cream.
- A Mermaid Sprinkles by Cool Mom with silver, purple, and blue colors for baking and decorating cupcakes and cakes.
- A Gold Sugar Crystals with 8 ounces by Sugar Deco.
- A Lieber's rainbow sprinkles with colorful jimmies for baking and decorating ice cream, 11 ounces.
- A Treasure Hunt Sprinkle Mix with colorful edible sprinkles for a birthday party in a 2oz size.

Current state. In the last subtask, SimpleMem and MAGE correctly purchased AmeriColor Tulip Red AmeriMist Airbrush Food Color (ASIN `B071S91WRZ`). Therefore, the correct transition for the current subtask is Red → Gold, and the ground-truth target is Gold Heart Sprinkles (ASIN `B08BFQ9B7T`) after comparing the price and rating between compatible options.

SimpleMem Trace

Retrieved memories.
- [Memories 1] Content: Betty Crocker Rich & Creamy Frosting - Cream Cheese - 16 oz - 3 pk with ... Topic: Final product purchase decision.

- [Memories 2] Content: The third available cake base option is a Duncan Hines Mermaid Cake Cup Vanilla Cake Mix with Sprinkles weighing 2.15 Oz, identified by ASIN B084JBT37C, priced at $6.99. ... Topic: Evaluation of Duncan Hines Mermaid Cake Cup.
- [Memories 3] Content: The shopping task for selecting and purchasing a compatible frosting was successfully completed. Topic: Task completion status.
- [Memories 4] Content: Strawberry cake toppers, lemon cupcake toppers, and lemon-flavored icing were excluded from consideration because ... Topic: Product option filtering.
- [Memories 5] Content: A search for Betty Crocker ready-to-serve cream cheese frosting in an 8 pack returned ... Topic: Second product search results.

Agent reasoning and actions.
<think> ... Is there any chance the previous product is Red or Pink? Memory 2 says "Duncan Hines Mermaid Cake Cup". Mermaid usually implies Blue/Silver/Green/Purple. The compatibility note says "Blue pairs well with Silver." This aligns perfectly. I will stick with this reasoning. ... </think> ... The previous product is likely the 'Duncan Hines Mermaid Cake Cup' (Blue), which pairs with Silver according to the rules. 'Mermaid Sprinkles' is the only option explicitly containing Silver. ...

Final purchase.
Mermaid Sprinkles by Cool Mom ... (B07YTFYGXL) at $13.95 with rating 4.8.

MAGE Trace

Compressed State.
- [Step 0] Product 1 Purchased: Gluten-Free Carrot Cake Mix. Price: $14.99. ASIN: B00T6NA7PA. Rating: 4.5.
- [Step 24] Product 2 Purchased: Betty Crocker Ready-to-Serve R & C Frost Cream Cheese - 8 Pack. Price: $34.27. ASIN: B003GQP06O. Rating: N.A..
- [Step 31] Product 3 Purchased: AmeriColor Tulip Red AmeriMist Airbrush Food Color, 9 oz. Price: $16.0. ASIN: B071S91WRZ. Rating: N.A..

Agent reasoning and actions.
... Prior purchase contains 'Red'. According to rules, Red pairs with Gold and avoids Green, Silver, Rainbow. Option 1 (Gold Heart Sprinkles) contains Gold -> compatible. Option 2 (Mermaid Sprinkles) contains silver -> incompatible. Option 3 (Gold Sugar Crystals) contains Gold -> compatible. Option 4 (Lieber's rainbow sprinkles) contains rainbow -> incompatible. Option 5 (Treasure Hunt Sprinkle Mix) has no conflict keywords -> compatible. I will search all options to check prices, then select the highest-priced compatible one within the remaining budget of $34.74. ...

Final purchase.
Gold Heart Sprinkles ... B08BFQ9B7T at $14.95 with rating 4.4.

Case analysis. SimpleMem assembles the agent context from semantically similar entries, which causes state fragmentation: stale frosting memories and a non-final Mermaid candidate replace the active execution state that Product 3 was AmeriColor Tulip Red. This breaks execution-state integrity, so the agent maps Mermaid to Blue/Silver/Green/Purple and follows the wrong Blue → Silver transition rather than the correct Red → Gold transition. In contrast, MAGE reads the current execution state from the active path, preserving the purchase chain and selecting Gold Heart Sprinkles for the right reason.

Case Study B: Mem0 Retrieves Candidate Memories Instead of the Final Purchase

Task. task_128_subtask_1, Buy Product 2, Select Matching Footrest:

Compatibility notes:
- Leather pairs well with Leather.
- Velvet pairs well with Velvet.

Avoid notes:
- Leather avoids Velvet.
- Velvet avoids Leather.

Available options:
- A blue PU leather ottoman storage chest with easy care for my living room.
- A B FSOBEIIALEO storage ottoman cube with velvet material in beige, 2 pack.
- A Best Master Furniture rectangle ottoman with grey velvet upholstery.
- A Modway faux leather armchair and ottoman set in gray.
- A GARO leather storage ottoman bench with collapsible design in black.

Current state. In the previous subtask, Mem0 and MAGE correctly purchased Blackjack Furniture Binion Leather Match Upholstered Modern Living Room Loveseat in Jet Black (ASIN B09H15V1KF) at $918.0 with rating 5. Therefore, the correct transition for the current subtask is Leather → Leather, and the ground-truth target is the blue PU leather Ottoman Storage Chest (ASIN B09M76K4F8) after comparing the prices among compatible leather options.

Mem0 Trace

Retrieved memories.
- [Memories 1] Content: The 91-inch reversible sectional sofa (B09PV5JQ91) is available in multiple sizes and colors including greyish, beige, blue, grey, and light grey.
- [Memories 2] Content: The second primary seating option the user is considering is a modern 3-seater velvet couch with a tufted back in light gray.
- [Memories 3] Content: User verified the first seating option (B09PV5JQ91) is priced at $889.00 with a 5-star rating and is now comparing it against the second option.
- [Memories 4] Content: User is evaluating primary seating options with the goal of purchasing the highest-priced item, using ratings as a tiebreaker for compatible choices.
- [Memories 5] Content: User searched for a modern 3-seater velvet couch with tufted back in light gray and found matching product B09H2XT45M, priced at $599.99.

Agent reasoning and actions.
... Based on the retrieved memories, the user's primary seating selection involves velvet furniture (modern 3-seater velvet couch B09H2XT45M and 91-inch reversible sectional sofa in velvet grey B09PV5JQ91). According to compatibility notes, Velvet pairs well with Velvet, and Velvet avoids Leather. Therefore, I need to select a velvet footrest. ... The other options (PU leather, faux leather, leather) are not compatible with velvet. ...

Final purchase.
Best Master Furniture SH003 Rectangle Ottoman, Grey Velvet ... (B08QYLGB9R) at $362.92.

MAGE Trace

Compressed State.
- [Step 0] Product 1 Purchased: Blackjack Furniture Binion Leather Match Upholstered Modern Living Room Loveseat, Jet Black. Price: $918.0. ASIN: B09H15V1KF. Rating: 5.

Agent reasoning and actions.
... Product 1 contains "Leather". Compatibility rules state Leather pairs well with Leather and Leather avoids Velvet. I will filter options to those containing leather and not velvet. Options 1, 4, and 5 match this criterion. ... Option 1 (PU leather): compatible, price $343.89. Option 2 (velvet): incompatible. Option 3 (velvet): incompatible. Option 4 (faux leather): compatible, price $271.34. Option 5 (leather): compatible, price $42.88. Among compatible options, Option 1 has the highest price. ...

Final purchase.
Ottoman Storage Chest Ottomans for Doorway, PU Leather Easy Care & Easy Clean Footrest (Blue) ... B09M76K4F8 at $343.89.

Case analysis. Mem0 retrieves semantically related memories about primary-seating candidates rather than the final purchased state. The retrieved context contains multiple velvet-related memories, but omits the active purchase Blackjack Leather Loveseat (ASIN B09H15V1KF). As a result, the agent reconstructs the previous state as Velvet and follows the wrong Velvet → Velvet transition, selecting a grey velvet ottoman and excluding the leather options. In contrast, MAGE reads the current execution state from the active path, preserves that Product 1 was Leather, and follows Leather → Leather to buy the correct PU leather ottoman.

F Ethical Considerations

This work studies memory mechanisms for LLM-based agents and is not tied to a specific high-stakes deployment. Nevertheless, improving long-horizon agent reliability may also strengthen dual-use applications, including automated information gathering or tool-use workflows that could be misused if deployed without safeguards. In practical deployments, execution histories may contain sensitive user or environment information, so systems based on MAGE should apply data minimization, access control, retention limits, and domain-specific safety checks before storing or reusing traces. Since our method does not train new foundation models and reduces token consumption during agent execution, its direct environmental impact is mainly associated with inference-time computation.