Remember the Decision, Not the Description: A Rate-Distortion Framework for Agent Memory

Mingxi Zou[1,2] Zhihan Guo[3] Langzhang Liang[1,2]
Zhuo Wang[2] Qifan Wang[5] Qingsong Wen[6]
Irwin King[3] Lizhen Qu[4,*] Zenglin Xu[1,2,*]

[1]Shanghai Academy of AI for Science (SAIS) [2]Fudan University
[3]The Chinese University of Hong Kong [4]Monash University
[5]Meta AI [6]AI Research Institute, Squirrel Ai Learning
[*]Corresponding authors

Abstract

Long-horizon language agents must operate under limited runtime memory, yet existing memory mechanisms often organize experience around descriptive criteria—relevance, salience, or summary quality. For an agent, however, memory is valuable not because it faithfully describes the past, but because it preserves the distinctions between histories that must remain separated under a fixed budget to support good decisions. We cast this as a decision-centric rate-distortion problem, measuring memory quality by the loss in achievable decision quality induced by compression. This yields an exact forgetting boundary for what can be safely forgotten, and a memory-distortion frontier characterizing the optimal tradeoff between memory budget and decision quality. Motivated by this decision-centric view of memory, we propose **DeMem**, an online memory learner that refines its partition only when data certify that a shared state would induce decision conflict, and prove near-minimax regret guarantees. On both controlled synthetic diagnostics and long-horizon conversational benchmarks, DeMem yields consistent gains under the same runtime budget, supporting the principle that memory should preserve the distinctions that matter for decisions, not descriptions.

1 Introduction

Large language model (LLM) agents increasingly operate over long interaction horizons. They must answer using earlier dialogue, plan from past observations, and adapt behavior using feedback accumulated across trials and sessions [43, 34]. Yet long-horizon interaction remains bottlenecked by memory. Increasing the context window alone does not guarantee reliable use of distant evidence [21], and recent benchmarks continue to expose failures in long-term conversational memory, incremental interaction, test-time learning, and constraint-consistent recall [22, 41, 11, 20].

These limitations have motivated a growing class of external and persistent memory mechanisms for language models and agents. Retrieval-augmented methods make past evidence available at generation time [18], while OS-style and explicit long-term memory systems maintain state beyond the current context window [27, 40]. More recent systems organize memory through graph-structured, self-organizing, or production-oriented memory layers [9, 3, 24, 16].

Existing agent memory systems typically decide what to store, retrieve, or compress using *descriptive* properties of experience. Retrieval-based methods prioritize semantic relevance to the current query [18], while long-term and OS-style memory systems often rely on salience, recency, or summary fidelity to maintain usable persistent state [27, 40]. Graph-structured and self-organizing memory systems further organize experience through entity links, temporal structure, or other relational cues [9, 3, 24]. Such criteria are useful for organizing past experience, but they do not directly capture the

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

downstream purpose of memory: supporting decisions under a strict runtime budget. This concern is not hypothetical. On the LoCoMo benchmark, we find that descriptive similarity is a *statistically significant but weak predictor of evidence compatibility*: the Spearman correlation is only $\rho = 0.103$ (AUC $= 0.548$). Under a matched answer-time budget, description-based retrieval recovers only 66% of gold evidence, compared to 83% for DeMem(ours). Among the queries where description-based memory fails, 85% of errors trace directly to evidence miss or dilution caused by this mismatch (Appendix E.13, Tables 12–14). Thus, what must be remembered should be defined by its effect on decision quality, not by its descriptive fidelity.

In this paper, we study budgeted agent memory through the lens of *decision-centric compression*. This view is consistent with recent calls for decision-centric design in LLM systems [37], but specializes that perspective to the memory bottleneck: at decision time, the agent cannot rely on the full interaction history, and must instead access the past through one of K internal memory states—a fixed number of memory slots available to the decision policy. The budget is therefore placed on runtime decision states, rather than on offline storage or preprocessing. Motivated by this view, we propose DEMEM, a budgeted memory method with K runtime slots. DeMem routes each history–query pair to a slot and refines the slots only when shared memory would create a decision conflict. In this way, it remembers distinctions that matter for action rather than description.

Our contributions answer three concrete questions raised by this view:

- **When can histories be safely merged? — A decision-centric theory of forgetting.** We formalize when histories can share memory without hurting downstream decisions, yielding an exact forgetting boundary separating safe merges from decision-conflicting ones, and a memory–distortion frontier describing the best achievable decision quality under a fixed memory budget.
- **How can decision-aware memory be learned online? — A certified online memory learner for decision-aware refinement.** We introduce DeMem, an online memory learner that maintains K runtime memory states, refines them only upon certified decision conflict, and comes with near-minimax regret guarantees.
- **Does this principle improve practical agent memory? — A practical K-slot agent memory system.** We instantiate DeMem with slot routing and a guarded conflict test (the illustrative DeMem workflow in Figure 1), and show consistent gains on synthetic diagnostics and long-horizon conversational memory benchmarks.

More broadly, our results suggest a simple principle for agent memory design: *remember the decision, not the description*.

2 Related Work

State abstraction and decision-preserving compression. Our work is theoretically closest to state abstraction and decision-preserving compression in sequential decision making, including model minimization, approximate aggregation, and policy abstraction[19, 8, 31, 1, 2, 48]. These works study when multiple states can be treated as equivalent for control, and characterize abstraction quality through value, model, or policy preservation. A closely related line develops behavioral metrics, including bisimulation-style distances for measuring when states are approximately interchangeable for decision making [6], while more recent work extends homomorphism ideas to continuous settings [32]. Our setting differs mainly in the object of compression. Prior work abstracts environment states or dynamics for planning, whereas we compress interaction histories into a limited set of runtime memory states and measure the resulting loss in decision quality.

Task-driven compression and bounded interaction. Our formulation is also related in spirit to task-driven information bottleneck approaches to estimation and control, which advocate preserving only the information needed for downstream decisions rather than for faithful reconstruction [26], as well as recent decision-centric perspectives on designing LLM systems around downstream utility rather than intermediate descriptive objectives [37]. A second nearby line studies control under bounded internal state, including finite-state controllers for partially observable settings [36] and, more recently, canonical abstractions induced by agent-bounded indistinguishability in bounded interaction [25]. We share the premise that capability constraints determine which distinctions matter, but provide concrete formal tools for the memory setting: an exact forgetting boundary, a memory–distortion frontier, and finite-sample regret guarantees for online refinement.

Memory systems and learned management for LLM agents. A large body of work augments language agents with external or persistent memory, including retrieval-based memory [18], long-term memory modules [40], operating-system-style or harness-managed memory [27, 14, 29], graph-structured or agentically organized memory [9, 42], production-oriented memory systems [3], and broader harness or externalization layers that manage context, tool use, persistent state, and long-horizon execution [28, 17, 51]. Another line studies learned or agentic memory control, including dynamic note organization, memory actions, RL-based long-/short-term memory management, atomic memory operations, self-evolving memory skills or architectures [42, 49, 44, 13, 47, 45], as well as memory–reasoning coupling, episodic memory, temporal memory selection, and generative latent memory for long-horizon or self-evolving agents [53, 35, 5, 46]. Recent surveys organize agent memory beyond the long-/short-term dichotomy, using taxonomies based on representation, function, and memory dynamics [50, 12, 4], while benchmarks show that robust long-term, incremental, and action-coupled memory remains difficult [22, 41, 11, 20, 38, 10, 33]. These works advance memory mechanisms, training procedures, and evaluations; our focus is complementary, characterizing the decision loss induced by bounded memory states and when histories can be safely merged.

3 Problem Setup and Fundamental Limits

This section develops the decision-centric memory formulation from the basic runtime bottleneck to its fundamental limits. We first formalize the K-state memory constraint, where the agent can access history only through a bounded internal state. Given this bottleneck, we then define the decision distortion induced by compression, which measures how much decision quality is lost when histories are merged. This loss notion allows us to characterize when merging is safe: Sections 3.3–3.4 derive an exact forgetting boundary and covering/packing measures for how many decision-distinguishable situations must remain separated. Finally, Section 3.5 shows that computing the optimal partition is computationally hard, motivating the online methods in Section 4.

3.1 Memory-constrained contextual decision model

We study a contextual decision problem under an explicit *runtime memory budget*, instantiated analytically as a contextual bandit. Let $\mathcal{H}$ be a finite set of dialogue histories, $\mathcal{Q}$ a finite set of queries, and $\mathcal{A}$ a finite set of downstream decisions with $|\mathcal{A}| = A$. A context is a pair $x = (h, q) \in \mathcal{X} \subseteq \mathcal{H} \times \mathcal{Q}$. In the conversational benchmarks of Section 5, h is the dialogue history, q is the evaluation query, and the action is the agent's generated answer. At each round $t = 1, \dots, T$: (i) a context $X_t = (H_t, Q_t) \sim \mathcal{D}$ is drawn i.i.d. from an unknown distribution $\mathcal{D}$ over $\mathcal{X}$; (ii) the learner chooses a decision $A_t \in \mathcal{A}$; (iii) the learner observes a reward $R_t \in [0, 1]$ with conditional mean

$$\mathbb{E}[R_t \mid X_t = x,\ A_t = a] \ = \ \mu(x, a), \tag{1}$$

where $\mu : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is fixed but unknown. The full-information oracle chooses $a^\star(x) \in \arg\max_{a \in \mathcal{A}} \mu(x, a)$ with value $\mu^\star(x) := \max_a \mu(x, a)$, and we measure regret

$$\mathrm{Reg}(T) \ := \ \sum_{t=1}^{T} \big(\mu^\star(X_t) - \mu(X_t, A_t)\big). \tag{2}$$

This i.i.d. sampling assumption is over answer-time decision instances, not over individual dialogue turns; we defer the modeling clarification to Appendix A.1.

Strict runtime memory constraint. At answer time, the current query Q_t is observed directly and is not counted against the memory budget. The bottleneck instead constrains access to the history: before acting, the learner maps the history into one of K runtime memory states through a query-aware encoder,

$$M_t = g_t(H_t, Q_t) \in [K], \qquad A_t = \pi_t(M_t, Q_t), \tag{3}$$

where $g_t : \mathcal{H} \times \mathcal{Q} \to [K]$ is a deterministic encoder and $\pi_t : [K] \times \mathcal{Q} \to \mathcal{A}$ is a deterministic decision rule. Thus, different queries may select different decision-relevant aspects of the same history, but the policy can act only from the current query and the selected memory state, not from the full history directly.

The resulting limits are query-conditional: for fixed q, we ask which histories can share a memory state without degrading decision quality. We define the static memory–distortion frontier over deterministic answer-time selectors, the class induced by deterministic decoding; Appendix A.3 separates this modeling choice from the exploration randomness used by the online learner. Proposition 5 makes the theory–implementation correspondence precise.

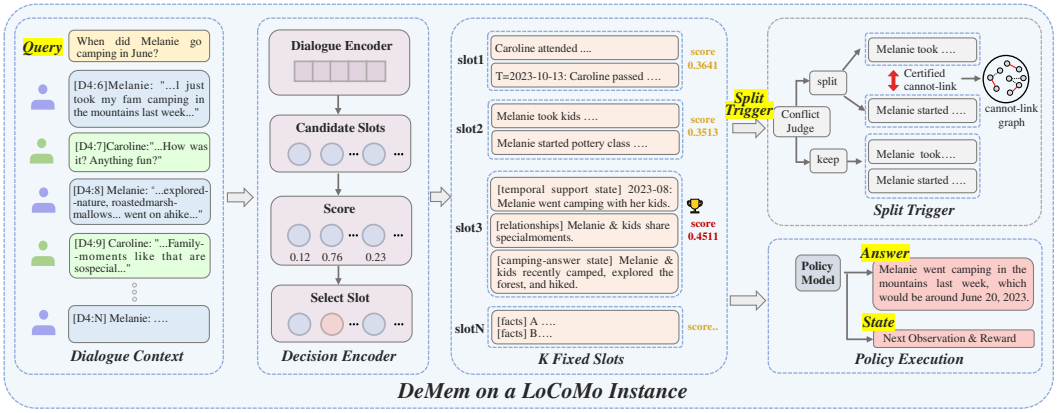


Figure 1: DeMem routes histories into bounded slots and refines them on decision conflict: statistical certificates in the online model and guarded conflict tests in the LLM implementation.

3.2 Decision distortion under memory

Having fixed the K-state memory interface, we now quantify what is lost when histories that may require different decisions are merged. Fix $q \in \mathcal{Q}$, and let $\mathcal{X}_q := \{h \in \mathcal{H} : (h, q) \in \mathcal{X}\}$ be the set of histories that may appear with query q. For each history–action pair (h, a), write $\mu_q(h, a) := \mu((h, q), a)$ and $\mu_q^\star(h) := \max_{a \in \mathcal{A}} \mu_q(h, a)$. Thus, $\mu_q^\star(h)$ is the value of the best action when the full history h is available for answering query q.

For any action a, define its suboptimality gap at history h as

$$\Delta_q(h, a) := \mu_q^\star(h) - \mu_q(h, a) \in [0, 1]. \tag{4}$$

A K-state memory encoder assigns each history to one of K compressed memory states. For fixed q, let $g_q : \mathcal{X}_q \to [K]$ denote the induced encoder $g_q(h) = g(h, q)$, and let $\pi_q : [K] \to \mathcal{A}$ denote the induced decision rule $\pi_q(m) = \pi(m, q)$. If history h is routed to state m, the resulting decision distortion is

$$d_q(h, m; \pi_q) := \mu_q^\star(h) - \mu_q\big(h, \pi_q(m)\big) = \Delta_q\big(h, \pi_q(m)\big). \tag{5}$$

This measures how much decision quality is lost because the policy sees only the compressed memory state m, rather than the full history h.

The best worst-case distortion achievable with K memory states for query q is

$$\epsilon_\infty^\star(K; q) := \inf_{g_q : \mathcal{X}_q \to [K]} \inf_{\pi_q : [K] \to \mathcal{A}} \sup_{h \in \mathcal{X}_q} d_q\big(h, g_q(h); \pi_q\big). \tag{6}$$

We call $\epsilon_\infty^\star(K; q)$ the decision rate–distortion frontier for query q: the smallest possible worst-case decision loss incurred by compressing histories into only K memory states. When the fixed query is clear from context, we write $\epsilon_\infty^\star(K)$.

3.3 Exact forgetting boundary

The distortion frontier tells us *how much* decision quality is lost under a memory budget; we now characterize *when* histories can be merged with decision loss at most ϵ.

Theorem 1 (Exact forgetting boundary)**.** *Fix a query q, $\epsilon \geq 0$ and a nonempty $C \subseteq \mathcal{X}_q$. The following are equivalent:*

1. *There exists $a \in \mathcal{A}$ with $\Delta_q(h, a) \leq \epsilon$ for all $h \in C$.*
2. *There exists a one-state encoder on C and a deterministic decision rule whose worst-case distortion over C is at most ϵ.*

A short proof appears in Appendix B.1. Within a fixed query fiber, the criterion is purely decision-theoretic: histories may share a memory state precisely when they admit a common ϵ-optimal action for that query—not when they are descriptively similar or have close reward vectors. Across different queries, the same abstract memory state may support different actions through the original selector $\pi(m, q)$.

Theorem 1 motivates a natural distance. Define the *pairwise decision distance*

$$d^q_{\mathrm{dec}}(h,h') \;\coloneqq\; \min_{a\in\mathcal{A}} \max\big\{\Delta_q(h,a),\ \Delta_q(h',a)\big\}, \tag{7}$$

the smallest ϵ for which h and h' can share a state at distortion ϵ. More generally, the *cluster decision radius* of $C \subseteq \mathcal{X}_q$ is

$$\rho^q_{\mathrm{dec}}(C) \;\coloneqq\; \min_{a\in\mathcal{A}} \max_{h\in C} \Delta_q(h,a),$$

the smallest worst-case loss from assigning a single action to all histories in C. Because d^q_{dec} need not satisfy the triangle inequality, the complexity measures below are defined through cluster radii directly.

3.4 Decision covering and packing bounds

Given the forgetting boundary and the decision distance, we can now determine the minimum number of memory states necessary to constrain the distortion within a target ϵ for the fixed query q. We introduce two combinatorial quantities. The decision covering number

$$\mathcal{N}^{\mathrm{dec}}_{\mathrm{cov}}(\epsilon;q) \;\coloneqq\; \min\Big\{K : \exists\, \mathcal{X}_q = \bigsqcup_{m=1}^{K} C_m \text{ s.t. } \max_m \rho^q_{\mathrm{dec}}(C_m) \le \epsilon\Big\} \tag{8}$$

is the fewest states that achieve worst-case distortion ϵ for query q. The decision packing number

$$\mathcal{N}^{\mathrm{dec}}_{\mathrm{pack}}(\epsilon;q) \;\coloneqq\; \max\Big\{|P| : P \subseteq \mathcal{X}_q,\ \forall h \neq h' \in P,\ d^q_{\mathrm{dec}}(h,h') > \epsilon\Big\} \tag{9}$$

is the largest set of mutually decision-incompatible histories at scale ϵ for query q.

Theorem 2 (Decision covering and packing bounds)**.** *For every $\epsilon > 0$:*

(1) if $K \ge \mathcal{N}^{\mathrm{dec}}_{\mathrm{cov}}(\epsilon;q)$, then $\epsilon^\star_\infty(K;q) \le \epsilon$; (2) if $\mathcal{N}^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon;q) > K$, then $\epsilon^\star_\infty(K;q) > \epsilon$.

Moreover, any stochastic memory variable M supporting worst-case distortion at most ϵ over the fixed query fiber must satisfy

$$I(Z;M) \ge \log \mathcal{N}^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon;q),$$

where Z is uniform over a maximum decision packing in $\mathcal{X}_q$.

Proofs are in Appendices B.2–B.4. The relevant notion of task entropy is the number of *decision-distinguishable* situations that must remain separable, not generic representational complexity. A distribution-dependent refinement replacing the worst-case frontier with average cluster radii is given in Appendix B.6 (Proposition 2); in Section 5 we directly measure its empirical counterpart.

3.5 Computational hardness of optimal memory

The covering and packing bounds determine the minimum budget at any target distortion, but can this optimal partition be found efficiently?

Theorem 3 (Computational hardness)**.** *Given $\mu \in [0,1]^{N\times A}$, an integer K, and a rational $0 \le \epsilon < 1$, deciding whether $\epsilon^\star_\infty(K) \le \epsilon$ is* NP*-complete.*

The proof is in Appendix B.5; this NP-hardness motivates DeMem: instead of solving the global optimal partition exactly, it learns a conservative decision-compatible partition from data through certified refinements.

4 Online Learning with Certified Memory Splits

We now return to the online setting, where $x = (h,q)$ denotes a full answer-time context, while compatibility is assessed within each fixed-query fiber. The frontier $\epsilon^\star_\infty(K)$ describes the best achievable compression quality, but it depends on the unknown reward function μ. We therefore need an online learner that simultaneously (i) estimates which contexts are decision-compatible, (ii) respects the hard K-state budget, and (iii) refines memory only when the data justify doing so. Our method, DeMem (**De**cision-aware **Mem**ory), is built around one principle:

split a memory state only when the data certify that some pair of contexts inside it cannot safely share a near-optimal action.

A full specification is given in Algorithm 3 (Appendix C.3), while Algorithm 1 summarizes the epoch-level execution flow and Figure 1 provides an intuitive worked example. We summarize the two key mechanisms below and then state the regret guarantees.

4.1 Certified incompatibility and budgeted encoding

Incompatibility certificates. For each observed context–action pair (x, a), let $\hat{\mu}_t(x, a)$ be the empirical mean, $n_t(x, a)$ the count. From these we construct lower certificates $\underline{d}_t(x, x')$ for the pairwise decision distance satisfying, with probability at least $1 - \delta$,

$$\underline{d}_t(x, x') \le d_{\mathrm{dec}}(x, x') \qquad \text{for all sampled pairs and all } t. \tag{10}$$

Whenever $\underline{d}_t(x, x') > \epsilon$, the data certify that x and x' cannot share any memory state whose cluster radius is at most ϵ. This yields a one-sided cannot-link relation.

Importantly, the converse is not assumed: the absence of pairwise cannot-link edges inside a cluster does not by itself certify that the cluster admits a single shared ϵ-optimal action. For this reason, the regret bound below is stated in terms of the realized cluster radius, certified by $\overline{\rho}_t(C)$, rather than in terms of pairwise graph independence alone. The full construction and validity guarantees appear in Appendix C.1. For one-pass LLM benchmarks, we instantiate this criterion with a candidate-restricted guarded conflict test. This one-pass test uses noisy scorer feedback and does not inherit the finite-sample certificate or regret guarantee of the repeated-feedback model; implementation details and calibration checks are provided in Appendix D.11.

Certification exploration. To make cannot-link certificates statistically testable, DeMem ensures a minimum amount of context-level exploration before relying on cluster-level UCB. For a resolution parameter $\gamma > 0$, each observed context–action pair is sampled until

$$n_t(x, a) \ge B_t(\gamma) \coloneqq \left\lceil \frac{8 \log(4NAt^2/\delta)}{\gamma^2} \right\rceil.$$

Once this coverage condition holds, the algorithm falls back to the cluster-level UCB rule. Appendix C.4 explains how this step separates certifiable positive-margin conflicts from conflicts that remain part of the realized compression error.

Budgeted encoder. At the beginning of each epoch, DeMem builds a cannot-link graph over observed contexts and applies a polynomial-time greedy coloring subroutine to obtain a feasible K-slot partition $\mathcal{P}_e$ (Algorithm 2, Appendix C.2). Its certified realized compression price is

$$\epsilon_e^{\mathrm{cert}} \coloneqq \max_{C \in \mathcal{P}_e} \overline{\rho}_{t_e}(C),$$

which upper-bounds the true cluster distortion on the high-probability confidence event. The partition is then frozen for the epoch, and DeMem runs UCB at the memory-state level. The full graph construction and frozen-epoch protocol are deferred to Appendix C.2.

4.2 Regret guarantees

Even if μ were known, a K-state memory induces an unavoidable approximation error; learning adds a second, statistical term. Let $N_T \coloneqq |\{X_1, \ldots, X_T\}| \le \min\{N, T\}$ be the number of distinct contexts observed.

Theorem 4 (Upper bound for DeMem)**.** *Let $\bar{\epsilon}_T^{\mathrm{cert}} \coloneqq \max_{e \le \lceil \log_2 T \rceil} \epsilon_e^{\mathrm{cert}}$, where $\epsilon_e^{\mathrm{cert}}$ is the certified compression price of the epoch-e partition defined in Section 4.1. Let $B_T(\gamma)$ denote the certification threshold $B_t(\gamma)$ evaluated at $t = T$. With probability at least $1 - \delta$,*

$$\mathrm{Reg}(T) \;\le\; T \cdot O\big(\bar{\epsilon}_T^{\mathrm{cert}}\big) \;+\; \tilde{O}\big(\sqrt{AKT}\big) \;+\; O\big(AN_T B_T(\gamma)\big). \tag{11}$$

The three terms correspond to realized compression error, statistical learning over K memory states, and context-level certification exploration; the last term is $\tilde{O}(AN_T/\gamma^2)$. This cost makes cannot-link certificates statistically valid, avoiding a pairwise-independence proxy for cluster compatibility. By Lemma 1, the certified compression prices upper-bound the true cluster radii on the high-probability event, so the regret is charged to the realized cluster-level distortion rather than to pairwise graph independence. A proof sketch is in Appendix C.5.

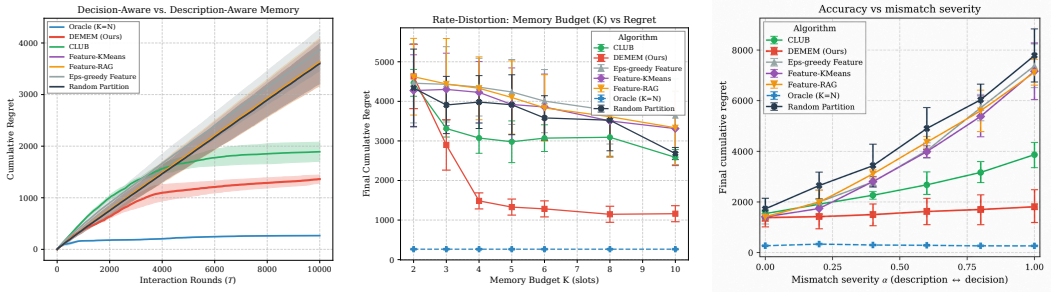
Theorem 5 (Minimax lower bound under K-state memory)**.** *There exists a universal constant $c > 0$ such that*

$$\inf_{\mathrm{alg}} \sup_{\mu, \mathcal{D}} \mathbb{E}[\mathrm{Reg}(T)] \;\ge\; c\,T\,\epsilon_\infty^\star(K) \;+\; c\sqrt{AKT}, \tag{12}$$

where the infimum ranges over algorithms whose answer-time decision rule is deterministic and whose access to the history is mediated by a K-state runtime memory.

Algorithm 1 DeMem in one loop

Initialize a K-slot partition $\mathcal{P}_0$ and context–action statistics
for each epoch e **do**
  Act: freeze $\mathcal{P}_e$ and route each context, $m_t \leftarrow g_{\mathcal{P}_e}(x_t)$
  choose a_t by exploration or slot-level UCB, $a_t \leftarrow \mathrm{Explore/UCB}(m_t, x_t)$; update statistics
  Certify: build cannot-link edges $E_e = \{(x, x') : \underline{d}_e(x, x') > \epsilon\}$
  Refresh: compute $\mathcal{P}_{e+1} \leftarrow \mathrm{GreedyColor}_K(E_e)$
end for



(a) Cumulative regret. (b) Memory Budget vs. Regret. (c) Mismatch mechanism analysis.

Figure 2: Synthetic results. (a) Cumulative regret on the Decoupled Bandit. (b) Memory–distortion tradeoff under varying K. (c) Performance under increasing description–decision mismatch.

A proof sketch is in Appendix C.6. Together, Theorems 4 and 5 show that DeMem matches the minimax statistical learning term up to logarithmic factors. The remaining excess term is the realized compression price

$$\bar{\epsilon}_T^{\mathrm{cert}} - \epsilon_\infty^\star(K),$$

together with the context-certification overhead. The greedy graph routine affects how the partition is found, but the regret bound is charged to the certified cluster radius of the returned partition. Block-stationary extensions are stated in Appendix C.7.

From theory to slot-based agent memory. A K-slot memory system directly realizes the runtime bottleneck of Section 3: the router implements g, the selected slot identity is M_t, and the policy conditions on the current query together with the selected memory state. The complete DeMem procedure is provided in Appendix C.3. Proposition 5 (Appendix D) decomposes the realized distortion into

$$\underbrace{\epsilon_\infty^\star(K)}_{\text{compression}} + \underbrace{\eta_{\mathrm{route}}}_{\text{routing}} + \underbrace{\eta_{\mathrm{read}}}_{\text{realization}} .$$

The complete DeMem procedure is given in Appendix C.3, and the theory–implementation correspondence is summarized in the alignment table of Appendix D.8. Appendix E.10 empirically validates the decomposition above, while Appendix E.12 shows that the same decision-aware selection criterion can improve other memory systems as a drop-in component.

5 Experiments

5.1 Experimental Setup

Synthetic environment. We use a *Decoupled Bandit* in which each context has a descriptive representation and a latent decision identity that are intentionally misaligned: descriptively similar contexts need not share the same optimal action. Baselines include an unconstrained **Oracle**, **CLUB**-style online clustering [7], description-aware methods (**Feature-KMeans**, **Feature-RAG**, ϵ-greedy feature clustering), and a **Random Partition** control. We report cumulative regret, an empirical memory–distortion curve (varying K), and a mismatch-severity sweep interpolating between aligned and misaligned regimes. Full definitions are in Appendix E.1.

benchmarks and baselines. We evaluate on **LoCoMo** [22], **LongMemEval** [41], and **MemoryArena** [10] against **FullContext**, **RAG** [18], **LangMem** [15], **Mem0** [3], **Zep** [30], **Nemori** [24], **EMem-G** [52], and **Mnemis** [39]. All methods share the same answering backbones (`gpt-4o-mini`, `gpt-4.1-mini`), deterministic decoding. Answers are scored by two held-out LLM judges,

Table 1: Results on LoCoMo. Mean ± std over runs.

Backbone	Method	Temporal	Open Domain	Multi-Hop	Single-Hop	Overall
GPT-4o-mini	FullContext	$0.515_{\pm 0.050}$	$0.428_{\pm 0.067}$	$0.629_{\pm 0.057}$	$0.791_{\pm 0.039}$	$0.681_{\pm 0.053}$
	LangMem	$0.208_{\pm 0.090}$	$0.423_{\pm 0.108}$	$0.470_{\pm 0.096}$	$0.565_{\pm 0.079}$	$0.464_{\pm 0.093}$
	Mem0	$0.453_{\pm 0.072}$	$0.365_{\pm 0.099}$	$0.552_{\pm 0.080}$	$0.644_{\pm 0.066}$	$0.570_{\pm 0.077}$
	RAG	$0.572_{\pm 0.101}$	$0.590_{\pm 0.126}$	$0.543_{\pm 0.120}$	$0.698_{\pm 0.108}$	$0.637_{\pm 0.115}$
	Zep	$0.545_{\pm 0.076}$	$0.336_{\pm 0.100}$	$0.457_{\pm 0.088}$	$0.592_{\pm 0.073}$	$0.541_{\pm 0.082}$
	Nemori	$0.656_{\pm 0.057}$	$0.401_{\pm 0.086}$	$0.596_{\pm 0.068}$	$0.787_{\pm 0.045}$	$0.701_{\pm 0.060}$
	EMem-G	$0.717_{\pm 0.053}$	$0.517_{\pm 0.078}$	$0.702_{\pm 0.059}$	$0.782_{\pm 0.042}$	$0.737_{\pm 0.051}$
	Mnemis	$0.878_{\pm 0.037}$	$0.793_{\pm 0.060}$	$0.798_{\pm 0.054}$	$\mathbf{0.938}_{\pm 0.028}$	$0.888_{\pm 0.042}$
	DeMem	$\mathbf{0.919}_{\pm 0.031}$	$\mathbf{0.868}_{\pm 0.043}$	$\mathbf{0.847}_{\pm 0.042}$	$0.932_{\pm 0.018}$	$\mathbf{0.911}_{\pm 0.040}$
GPT-4.1-mini	FullContext	$0.692_{\pm 0.057}$	$0.511_{\pm 0.081}$	$0.725_{\pm 0.053}$	$0.831_{\pm 0.040}$	$0.763_{\pm 0.058}$
	LangMem	$0.461_{\pm 0.087}$	$0.531_{\pm 0.080}$	$0.659_{\pm 0.068}$	$0.800_{\pm 0.046}$	$0.687_{\pm 0.063}$
	Mem0	$0.514_{\pm 0.077}$	$0.428_{\pm 0.093}$	$0.629_{\pm 0.066}$	$0.669_{\pm 0.062}$	$0.614_{\pm 0.071}$
	RAG	$0.710_{\pm 0.110}$	$0.634_{\pm 0.117}$	$0.593_{\pm 0.103}$	$0.727_{\pm 0.113}$	$0.693_{\pm 0.106}$
	Zep	$0.554_{\pm 0.076}$	$0.383_{\pm 0.097}$	$0.495_{\pm 0.084}$	$0.623_{\pm 0.069}$	$0.570_{\pm 0.073}$
	Nemori	$0.731_{\pm 0.056}$	$0.454_{\pm 0.086}$	$0.706_{\pm 0.059}$	$0.811_{\pm 0.042}$	$0.753_{\pm 0.051}$
	EMem-G	$0.757_{\pm 0.047}$	$0.660_{\pm 0.064}$	$0.747_{\pm 0.052}$	$0.872_{\pm 0.031}$	$0.812_{\pm 0.040}$
	Mnemis	$0.891_{\pm 0.031}$	$0.858_{\pm 0.034}$	$0.805_{\pm 0.046}$	$\mathbf{0.940}_{\pm 0.025}$	$0.906_{\pm 0.039}$
	DeMem	$\mathbf{0.921}_{\pm 0.039}$	$\mathbf{0.908}_{\pm 0.040}$	$\mathbf{0.831}_{\pm 0.048}$	$0.935_{\pm 0.037}$	$\mathbf{0.920}_{\pm 0.038}$

`gpt-4o-mini` and `gpt-4.1-mini`(binary on LoCoMo, graded on LongMemEval); reported numbers average across judges and instances. A human-agreement study on 150 stratified LoCoMo instances confirms that the LLM judges align well with human majority vote ($\kappa = 0.79$, 91.3% agreement), with no systematic bias toward any method (Appendix E.14). Details on preprocessing, prompting, baseline instantiation, budget matching, and judge prompts are in Appendix E.2.

5.2 Synthetic Results

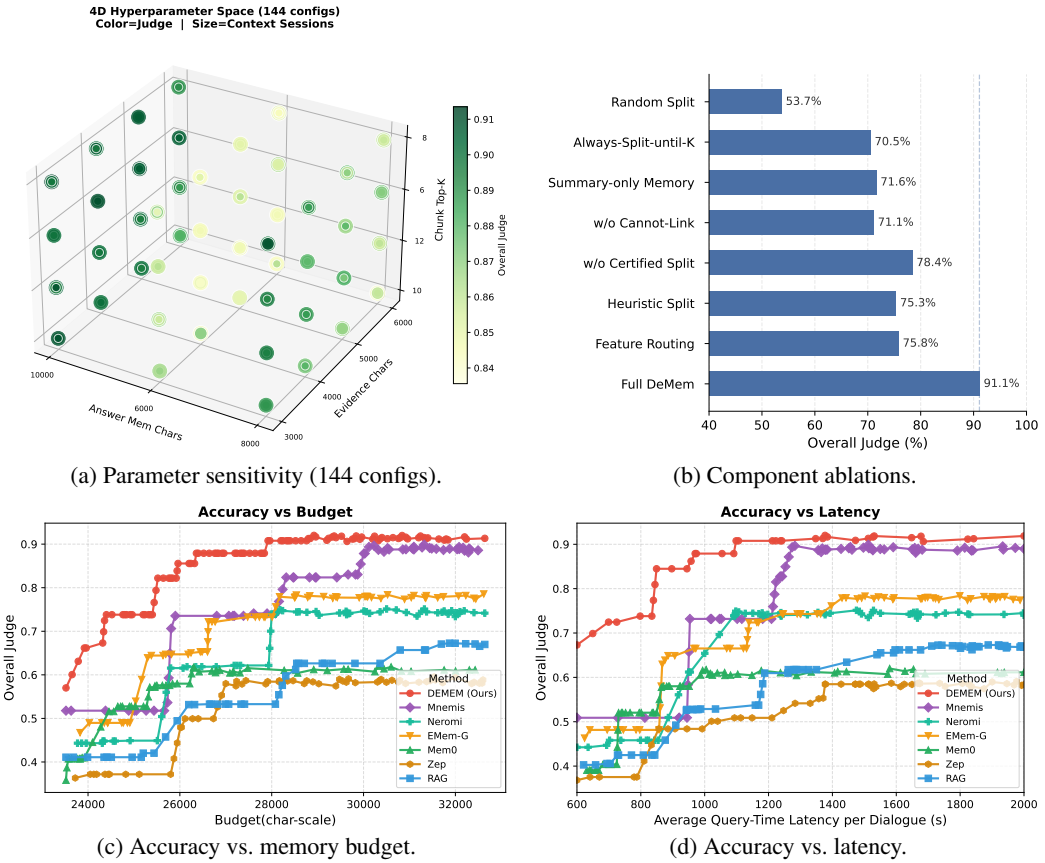
Figure 2 summarizes the synthetic results. DeMem achieves the lowest cumulative regret among all budgeted methods (a), attains a more favorable memory–distortion tradeoff (b), and widens its advantage as the mismatch between descriptive and decision similarity grows (c). Description-aware baselines merge contexts that require different actions, leading to persistent decision error; DeMem avoids this by refining memory only on certified decision conflict. We additionally measure the induced partition distortion (Appendix E.1.1, Figure 4): DeMem consistently learns the lowest-distortion partition, and downstream regret is broadly monotone in this quantity across methods and seeds, confirming the mechanism predicted by the theory. Full environment definitions and baseline specifications are in Appendix E.1.

5.3 Main Results on LoCoMo and LongMemEval

Table 1 reports results on **LoCoMo**. DeMem achieves the highest mean overall score on both backbones. The pattern is especially clear on Temporal and Open Domain questions, where preserving distinctions across distant interactions is important. On LongMemEval (Table 4, Appendix E.3), DeMem again obtains the best mean overall score on both backbones, with the largest gains on categories requiring cross-session integration. DeMem also lies on a favorable accuracy–resource frontier (Appendix E.5, Figure 3), characterizing answer-time use of the fixed memory budget. End-to-end construction and feedback costs are accounted for separately in Section 5.5. Results under Llama-3.1-70B (Appendix E.7) confirm that the advantage is not tied to a specific backbone family and generalizes across both proprietary and open-weight models. Mechanism audits on real benchmarks show the same pattern: DeMem forms purer, less conflicted memory states and yields partitions closer to oracle support than non-decision-aware alternatives (Appendices E.8 and E.9). Results on the agentic MemoryArena benchmark [10] show the same pattern: DeMem achieves the highest mean average success rate among the compared methods (Appendix E.4).

5.4 Ablation and Robustness

We ablate splitting, routing, and memory representation (variant definitions in Appendix E.6). Figure 3(b) shows that all ablations degrade performance. Removing the guarded split test is damaging: heuristic, aggressive, and random split variants all underperform the guarded split rule. Guarded splits are also selective and accurate, firing on only 4.6% of LoCoMo routing events while reaching



(a) Parameter sensitivity (144 configs).

(b) Component ablations.

(c) Accuracy vs. memory budget.

(d) Accuracy vs. latency.

Figure 3: Ablation and robustness. (a) Parameter sensitivity over 144 configurations. (b) Component ablations (defined in Appendix E.6); the plotted label *w/o Certified Split* denotes removal of the practical guarded conflict test. (c) Accuracy vs. memory budget. (d) Query-time latency; full pipeline costs are reported separately in Appendix H.1.

85% precision against gold annotations (Appendix E.11). Feature-based routing and description-only summaries further reduce performance, supporting the need to preserve decision-relevant distinctions throughout the memory pipeline.

Additional diagnostics support the same mechanism. On an annotated LoCoMo subset, the observed answer-level distortion reaches about 81% of the predicted bridge upper bound, with routing as the largest implementation contributor (Appendix E.10). A theory-faithful DeMem-Core variant retaining only slot structure, routing, and guarded refinement still reaches 90.8 overall (Appendix D.10). Under corrupted LoCoMo feedback and against feedback-trained or memory-action baselines, DeMem remains strongest overall (Appendices E.16 and E.17, Table 19). Overall, the split-precision, ablation, and DeMem-Core results are consistent with decision-aware state refinement being a key contributor to DeMem's gains.

5.5 Cost, Paired Reliability, and Backbone Generalization

End-to-end cost and reuse. On LoCoMo with GPT-4o-mini, we count all method-internal LLM calls and input/output tokens, including construction, maintenance, counterfactual generation, feedback scoring, and answering. The common final evaluation judge is excluded for every method. DeMem averages 4.0 calls, 9.6k tokens, and 24.8 s per query, compared with 1.0 call, 13.2k tokens, and 21.0 s for longer-context inference. At a matched total budget of 9.6k tokens, DeMem scores 0.911 versus 0.674 for longer context. This efficiency depends on reuse: one-off DeMem costs 34.6k tokens and 71.4 s, and token-cost parity is reached after approximately seven queries reuse the maintained memory. Appendix H.1 reports the full breakdown.

Paired uncertainty. Dialogue/source-history clusters are kept intact in 10,000 paired bootstrap resamples and paired permutation tests. Across the eight predefined DeMem–Mnemis and DeMem–EMem-G comparisons on two datasets and two backbones, seven remain significant at the 0.05 level after Holm correction. The LoCoMo/GPT-4.1-mini comparison with Mnemis has a positive gain of 0.014 but a 95% interval of $[-0.007, 0.035]$ and adjusted $p = 0.109$; its advantage is inconclusive. Full intervals and adjusted values are in Appendix H.2.

Additional answering backbones. On LongMemEval, DeMem improves over Mnemis from 0.852 to 0.870 with Qwen3.5-9B, from 0.899 to 0.922 with Qwen3.5-122B-A10B, and from 0.914 to 0.931 with GPT-5.4 mini. Each comparison matches the split, prompts, decoding settings, answering model, and answer-time memory budget. Reasoning is disabled for GPT-5.4 mini and both Qwen models run in non-thinking mode. These are additional point-estimate comparisons; Appendix H.3 gives their protocol and results.

6 Conclusion

We introduced a decision-centric framework for budgeted agent memory, viewing runtime memory as a compression bottleneck whose quality is measured by downstream decision loss. This formulation yields an exact forgetting boundary, a memory–distortion frontier, and complexity measures for the decision-relevant distinctions that must be retained. Building on this formulation, the online DeMem algorithm refines memory under statistically certified decision conflict and enjoys regret guarantees that separate unavoidable compression loss from statistical learning. Its LLM implementation uses a candidate-restricted guarded conflict test. Empirically, DeMem improves performance under fixed memory budgets, supporting decision-centric memory organization. Overall, agent memory should preserve the distinctions that matter for action, not merely description. At the same time, DeMem is best viewed as a budgeted decision layer that complements richer descriptive memory stores; implementation considerations and broader impacts are discussed in Appendices F and G.

References

[1] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction, 2017. URL https://arxiv.org/abs/1701.04113.

[2] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

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

[4] Pengfei Du. Memory for autonomous LLM agents:mechanisms, evaluation, and emerging frontiers, 2026. URL https://arxiv.org/abs/2603.07670.

[5] Yiming Du, Baojun Wang, Yifan Xiang, Zhaowei Wang, Wenyu Huang, Boyang XUE, Bin Liang, Xingshan Zeng, Fei Mi, Haoli Bai, Lifeng Shang, Jeff Z. Pan, Yuxin Jiang, and Kam-Fai Wong. Memory-t1: Reinforcement learning for temporal reasoning in multi-session agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vQf2YR2Kpd.

[6] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011. doi: 10.1137/10080484X. URL https://doi.org/10.1137/10080484X.

[7] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits, 2014. URL https://arxiv.org/abs/1401.8257.

[8] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(02)00376-4. URL https://www.sciencedirect.com/science/article/pii/S0004370202003764. Planning with Uncertainty and Incomplete Information.

[9] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models, 2025. URL https://arxiv.org/abs/2405.14831.

[10] 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, 2026. URL https://arxiv.org/abs/2602.16313.

[11] Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DT7JyQC3MR.

[12] Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of ai agents, 2026. URL https://arxiv.org/abs/2512.13564.

[13] Yupeng Huo, Yaxi Lu, Zhong Zhang, Haotian Chen, and Yankai Lin. Atommem : Learnable dynamic agentic memory with atomic memory operation, 2026. URL https://arxiv.org/abs/2601.08323.

[14] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

[15] LangChain AI. Langmem: Long-term memory sdk for LLM agents, 2024. URL https://github.com/langchain-ai/langmem.

[16] 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.

[17] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

[18] 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, 2021. URL https://arxiv.org/abs/2005.11401.

[19] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In *AI&M*, 2006. URL https://api.semanticscholar.org/CorpusID:245037.

[20] Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. Locomo-plus: Beyond-factual cognitive memory evaluation framework for LLM agents, 2026. URL https://arxiv.org/abs/2602.10715.

[21] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

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

[23] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, July 1983. ISSN 0004-5411. doi: 10.1145/2402.322385. URL https://doi.org/10.1145/2402.322385.

[24] Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science, 2025. URL https://arxiv.org/abs/2508.03341.

[25] Anthony T. Nixon. The myhill-nerode theorem for bounded interaction: Canonical abstractions via agent-bounded indistinguishability, 2026. URL https://arxiv.org/abs/2603.21399.

[26] Vincent Pacelli and Anirudha Majumdar. Task-driven estimation and control via information bottlenecks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2061–2067, 2019. doi: 10.1109/ICRA.2019.8794213.

[27] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards LLMs as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

[28] Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-language agent harnesses, 2026. URL https://arxiv.org/abs/2603.25723.

[29] Mofasshara Rafique and Laurent Bindschaedler. Clawvm: Harness-managed virtual memory for stateful tool-using LLM agents. 2026. URL https://api.semanticscholar.org/CorpusID:287432731.

[30] 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.

[31] Balaraman Ravindran and Andrew G Barto. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes. 2004.

[32] Sahand Rezaei-Shoshtari, Rosie Zhao, Prakash Panangaden, David Meger, and Doina Precup. Continuous mdp homomorphisms and homomorphic policy gradient, 2022. URL https://arxiv.org/abs/2209.07364.

[33] Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2actbench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents, 2026. URL https://arxiv.org/abs/2601.19935.

[34] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[35] Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. REMem: Reasoning with episodic memory in language agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fugnQxbvMm.

[36] Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe policy improvement for pomdps via finite-state controllers, 2023. URL https://arxiv.org/abs/2301.04939.

[37] Wei Sun. Decision-centric design for LLM systems. 2026. URL https://api.semanticscholar.org/CorpusID:287021773.

[38] Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of LLM-based agents, 2025. URL https://arxiv.org/abs/2506.21605.

[39] 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, 2026. URL https://arxiv.org/abs/2602.15313.

[40] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 74530–74543. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ebd82705f44793b6f9ade5a669d0f0bf-Paper-Conference.pdf.

[41] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

[42] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents, 2025. URL https://arxiv.org/abs/2502.12110.

[43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[44] Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents, 2026. URL https://arxiv.org/abs/2601.01885.

[45] Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems, 2025. URL https://arxiv.org/abs/2512.18746.

[46] Guibin Zhang, Muxin Fu, and Shuicheng YAN. Memgen: Weaving generative latent memory for self-evolving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vI56m4Iu4e.

[47] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026. URL https://arxiv.org/abs/2602.02474.

[48] Min Zhang, Hongyao Tang, Jianye Hao, and Yan Zheng. Towards a unified policy abstraction theory and representation learning approach in markov decision processes, 2022. URL https://arxiv.org/abs/2209.07696.

[49] Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2510.12635.

[50] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents, 2024. URL https://arxiv.org/abs/2404.13501.

[51] Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering, 2026. URL https://arxiv.org/abs/2604.08224.

[52] 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.

[53] Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

A Additional Preliminaries and Technical Reductions

A.1 Modeling details for answer-time decision instances

The i.i.d. assumption in Section 3.1 is made over answer-time decision instances $X_t = (H_t, Q_t)$, rather than over the individual turns inside a dialogue. A history H_t may itself contain temporally dependent, multi-session interactions. The contextual-bandit abstraction treats such histories as sampled decision contexts in order to isolate the runtime memory bottleneck faced by the agent at answer time.

The abstraction is made at the level of answer-time decision instances rather than individual dialogue turns. Each instance may contain temporally dependent, multi-session history, while the agent acts using the current query and a bounded runtime memory state summarizing the relevant past. This isolates the answer-time memory bottleneck analyzed in the main text.

A.2 Concentration

We repeatedly use the following standard fact: if $\hat{\mu}$ is the empirical mean of n i.i.d. rewards in $[0, 1]$ with mean μ, then for any $\delta \in (0, 1)$,

$$\Pr\left(|\hat{\mu} - \mu| \geq \sqrt{\frac{\log(2/\delta)}{2n}}\right) \leq \delta. \tag{13}$$

A.3 Deterministic decision rules

The memory-distortion frontier in the main text is defined over deterministic encoders and deterministic decision rules. This restriction is intentional. The language-agent implementation uses deterministic decoding: given a query and a selected memory slot, the answer string is fixed. Randomness used by the online learner for exploration is part of the learning protocol and is not included in the static memory-compression frontier.

Proposition 1 (Deterministic reduction under deterministic decoding)**.** *Fix a query q, an answering backbone F with deterministic decoding, a prompt template $\mathcal{P}$, and a K-slot memory system with slot contents $\{S_1, \ldots, S_K\}$. For each memory state m, define*

$$a_m(q) \coloneqq F\big(\mathcal{P}(q, S_m)\big).$$

Then the induced answer-time decision rule is a deterministic selector $m \mapsto a_m(q)$. Consequently, the distortion objectives in the main text optimize over deterministic per-state actions.

Proof. Deterministic decoding makes F a function rather than a stochastic map. For fixed q and fixed slot content S_m, the prompt $\mathcal{P}(q, S_m)$ therefore produces a unique answer string $a_m(q)$. Once the encoder selects $m = g(h, q)$, the downstream action is fixed. Hence the induced policy is deterministic at answer time. □

Remark 1. The deterministic frontier matches the answer-time interface used in our agent implementation. With deterministic decoding, a fixed query and selected memory slot induce a fixed generated answer. Randomness used for online exploration is part of the learning protocol, whereas the static memory–distortion frontier characterizes the deployed deterministic answer-time selector.

B Proofs for the Decision Rate–Distortion Frontier

B.1 Proof of the exact forgetting boundary

Proof of Theorem 1. $(1) \Rightarrow (2)$. Suppose action $a \in \mathcal{A}$ satisfies $\Delta_q(h, a) \leq \epsilon$ for every $h \in C$. By Proposition 1, a one-state memory encoder on C may be paired with the deterministic decision rule that always plays a. The worst-case distortion over C is then

$$\max_{h \in C} \Delta_q(h, a) \leq \epsilon.$$

$(2) \Rightarrow (1)$. Given a one-state encoder on C with worst-case distortion at most ϵ, Proposition 1 implies the decision rule may be taken deterministic, selecting some fixed $a \in \mathcal{A}$. Then

$$\max_{h \in C} \Delta_q(h, a) \leq \epsilon,$$

which is exactly condition (1). □

Remark 2 (Equivalence with cluster radius). Condition (1) of Theorem 1 is equivalent to $\rho^q_{\mathrm{dec}}(C) \leq \epsilon$, since by definition

$$\rho^q_{\mathrm{dec}}(C) = \min_{a \in \mathcal{A}} \max_{h \in C} \Delta_q(h, a).$$

Thus the forgetting boundary admits three equivalent readings: the existence of a common ϵ-optimal action, the cluster radius bound $\rho^q_{\mathrm{dec}}(C) \leq \epsilon$, and the one-state distortion bound.

Corollary 1 (Pairwise forgetting boundary)**.** *Two histories $h, h' \in \mathcal{X}_q$ can be mapped to the same memory state with worst-case loss at most ϵ if and only if $d^q_{\mathrm{dec}}(h, h') \leq \epsilon$.*

Proof. Apply Theorem 1 to $C = \{h, h'\}$. Condition (1) requires an action a with

$$\max\{\Delta_q(h, a), \Delta_q(h', a)\} \leq \epsilon.$$

Minimizing over a gives

$$\min_{a \in \mathcal{A}} \max\{\Delta_q(h, a), \Delta_q(h', a)\} = d^q_{\mathrm{dec}}(h, h') \leq \epsilon,$$

where the equality is the definition (7). □

B.2 Proof of the covering upper bound

Proof of Theorem 2, part (i). By definition of $\mathcal{N}^{\mathrm{dec}}_{\mathrm{cov}}(\epsilon; q)$, there exists a partition

$$\mathcal{X}_q = \bigsqcup_{m=1}^{K} C_m \qquad \text{such that} \qquad \max_m \rho^q_{\mathrm{dec}}(C_m) \leq \epsilon.$$

For each cluster C_m, choose

$$a_m \in \arg\min_{a \in \mathcal{A}} \max_{h \in C_m} \Delta_q(h, a).$$

Map each $h \in C_m$ to state m and play action a_m. By Theorem 1, the worst-case distortion on each cluster is at most ϵ, so the overall worst-case distortion is at most ϵ. Therefore $\epsilon^\star_\infty(K; q) \leq \epsilon$. □

B.3 Proof of the packing lower bound

Proof of Theorem 2, part (ii). Let $P \subseteq \mathcal{X}_q$ be a set of size $K + 1$ such that

$$d^q_{\mathrm{dec}}(h, h') > 2\epsilon \qquad \text{for all distinct } h, h' \in P.$$

Any encoder $g_q : \mathcal{X}_q \to [K]$ must map two distinct elements $h, h' \in P$ to the same memory state. Suppose that state chooses action a. Then, by definition of d^q_{dec},

$$\max\{\Delta_q(h, a), \Delta_q(h', a)\} \geq d^q_{\mathrm{dec}}(h, h') > 2\epsilon.$$

Hence in particular the worst-case distortion over $\{h, h'\}$ exceeds ϵ. Therefore no K-state memory can achieve worst-case distortion at most ϵ, and thus $\epsilon^\star_\infty(K; q) > \epsilon$. □

B.4 Information lower bound

Proof of the information lower bound. Let P be a maximum decision packing at scale 2ϵ in $\mathcal{X}_q$, so that

$$|P| = \mathcal{N}^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon; q),$$

and let Z be uniform on P. Consider any stochastic memory variable M used by a policy whose decision depends only on M. Suppose that M supports worst-case distortion at most ϵ.

We first show that the conditional supports of M must separate the points in P. Indeed, if there exist distinct $h, h' \in P$ and a memory state m such that

$$\Pr(M = m \mid Z = h) > 0 \quad \text{and} \quad \Pr(M = m \mid Z = h') > 0,$$

then the policy must take the same action $a(m)$ after observing m for both h and h'. Since P is a decision packing at scale 2ϵ, we have

$$d^q_{\mathrm{dec}}(h, h') > 2\epsilon.$$

Therefore, for every action a,

$$\max\{\Delta_q(h, a), \Delta_q(h', a)\} > 2\epsilon > \epsilon.$$

In particular this holds for $a = a(m)$, contradicting the assumed worst-case distortion bound.

Hence the realized memory state M identifies the history Z within P: there exists a decoder ϕ such that

$$Z = \phi(M) \quad \text{almost surely.}$$

Thus $H(Z \mid M) = 0$. Since Z is uniform on P,

$$I(Z; M) = H(Z) - H(Z \mid M) = H(Z) = \log |P| = \log \mathcal{N}^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon; q).$$

□

B.5 Proof of computational hardness

Proof of Theorem 3. We reduce from SET COVER. An instance of SET COVER consists of a universe

$$\mathcal{U} = \{u_1, \ldots, u_N\},$$

a collection of subsets

$$\mathcal{S} = \{S_1, \ldots, S_A\}, \qquad S_j \subseteq \mathcal{U},$$

and an integer K. The question is whether there exist at most K sets whose union covers $\mathcal{U}$. We assume without loss of generality that every element belongs to at least one set; otherwise the instance is immediately infeasible.

Construct a memory instance as follows. The context set is the universe,

$$\mathcal{X} = \mathcal{U},$$

and the action set contains one action a_j for each set S_j. Define the reward matrix by

$$\mu(u_i, a_j) = \begin{cases} 1, & u_i \in S_j, \\ 0, & u_i \notin S_j. \end{cases}$$

Since every u_i belongs to at least one set, we have $\mu^\star(u_i) = 1$ for all u_i. Hence the suboptimality gap is

$$\Delta(u_i, a_j) = \mu^\star(u_i) - \mu(u_i, a_j) = \begin{cases} 0, & u_i \in S_j, \\ 1, & u_i \notin S_j. \end{cases}$$

Fix any rational $0 \le \epsilon < 1$. For any nonempty cluster $C \subseteq \mathcal{X}$, its decision radius is

$$\rho_{\mathrm{dec}}(C) = \min_{a_j \in \mathcal{A}} \max_{u_i \in C} \Delta(u_i, a_j).$$

Because all gaps are either 0 or 1, and $\epsilon < 1$, we have

$$\rho_{\mathrm{dec}}(C) \le \epsilon \quad \iff \quad \exists j \in [A] \text{ such that } C \subseteq S_j.$$

Thus a cluster has distortion at most ϵ exactly when all of its contexts are covered by a single set action.

We now prove equivalence.

First, suppose the SET COVER instance has a cover $S_{j_1}, \ldots, S_{j_K}$. Assign each element u_i to one selected set that contains it, and let C_ℓ be the elements assigned to S_{j_ℓ}. Then each $C_\ell \subseteq S_{j_\ell}$, so choosing action a_{j_ℓ} for cluster C_ℓ gives $\rho_{\mathrm{dec}}(C_\ell) = 0 \le \epsilon$. Hence $\epsilon^\star_\infty(K) \le \epsilon$.

Conversely, suppose $\epsilon^\star_\infty(K) \le \epsilon$. Then there exists a partition

$$\mathcal{X} = \bigsqcup_{m=1}^{K} C_m$$

and actions a_{j_m} such that

$$\max_{u_i \in C_m} \Delta(u_i, a_{j_m}) \le \epsilon \qquad \text{for every } m.$$

Since $\epsilon < 1$ and the gaps are binary, this implies $\Delta(u_i, a_{j_m}) = 0$ for every $u_i \in C_m$, hence $C_m \subseteq S_{j_m}$. Therefore the sets

$$S_{j_1}, \dots, S_{j_K}$$

cover all elements of $\mathcal{U}$. Thus the original SET COVER instance is feasible.

Therefore, deciding whether $\epsilon^\star_\infty(K) \le \epsilon$ is NP-hard for every rational $0 \le \epsilon < 1$. Membership in NP is immediate: a certificate consists of a partition $\mathcal{X} = \bigsqcup_{m=1}^{K} C_m$ and one action a_m per cluster, and verification only requires checking

$$\max_{x \in C_m} \Delta(x, a_m) \le \epsilon$$

for all m, which is polynomial in N, A, and K. Hence the problem is NP-complete. □

Remark 3. The reduction is stated directly in terms of cluster decision radius rather than pairwise decision distance. This is important because pairwise compatibility need not compose into cluster compatibility. The hardness result therefore applies to the true optimal memory-partition problem: even with complete knowledge of μ, finding a K-state partition whose clusters each admit a shared low-distortion action is NP-hard. Consequently, polynomial-time memory learners such as DeMem must use tractable relaxations and are evaluated by the realized cluster-level distortion of the partitions they return.

B.6 Average distortion and distribution-dependent floor

The main text focuses on the worst-case frontier $\epsilon^\star_\infty(K; q)$. This appendix records the corresponding average-distortion analogue, which is useful when the conditional history distribution is non-uniform. Fix a query q, and let $\mathcal{D}_q$ denote the distribution over histories in $\mathcal{X}_q$.

For an encoder–policy pair (g_q, π_q), define the average decision distortion

$$D_q(g_q, \pi_q) \;\coloneqq\; \mathbb{E}_{H \sim \mathcal{D}_q}\big[d_q(H, g_q(H); \pi_q)\big]. \tag{14}$$

The corresponding average-case frontier is

$$\epsilon^\star(K; q) \;\coloneqq\; \inf_{g_q : \mathcal{X}_q \to [K]} \inf_{\pi_q : [K] \to \mathcal{A}} D_q(g_q, \pi_q). \tag{15}$$

When $\mathcal{D}$ is non-uniform, $\epsilon^\star(K)$ can be substantially smaller than the worst-case frontier $\epsilon^\star_\infty(K)$. The following proposition, referenced in Section 3.4, makes this precise.

Proposition 2 (Distribution-dependent distortion floor)**.** *For any encoder $g_q : \mathcal{X}_q \to [K]$ and any decision rule $\pi_q : [K] \to \mathcal{A}$,*

$$D_q(g_q, \pi_q) \;\ge\; \sum_{m=1}^{K} \Pr_{\mathcal{D}_q}\big(g_q(H) = m\big)\; \rho^{\mathrm{avg},q}_{\mathrm{dec}}\big(g_q^{-1}(m)\big), \tag{16}$$

where

$$\rho^{\mathrm{avg},q}_{\mathrm{dec}}(C) \coloneqq \min_{a \in \mathcal{A}} \mathbb{E}[\Delta_q(Z, a) \mid Z \in C]$$

is the average cluster decision radius for query q.

Proof. Fix any encoder $g_q : \mathcal{X}_q \to [K]$ and any decision rule $\pi_q : [K] \to \mathcal{A}$. By Proposition 1, we may assume without loss of generality that $\pi_q(m)$ selects a single action $a_m \in \mathcal{A}$ for each $m \in [K]$.

Decompose the average distortion by conditioning on the memory state:

$$D_q(g_q, \pi_q) = \mathbb{E}_{Z\sim\mathcal{D}_q}\big[\mu_q^\star(Z) - \mu_q(Z, a_{g_q(Z)})\big]$$
$$= \sum_{m=1}^{K} \Pr_{\mathcal{D}_q}\big(g_q(Z) = m\big)\ \mathbb{E}\big[\Delta_q(Z, a_m) \mid g_q(Z) = m\big]. \tag{17}$$

For each cluster m, the action a_m is a specific element of $\mathcal{A}$. Therefore

$$\mathbb{E}\big[\Delta_q(Z, a_m) \mid g_q(Z) = m\big] \ \geq\ \min_{a\in\mathcal{A}} \mathbb{E}\big[\Delta_q(Z, a) \mid g_q(Z) = m\big] \ =\ \rho_{\mathrm{dec}}^{\mathrm{avg},q}\big(g_q^{-1}(m)\big). \tag{18}$$

Substituting (18) into (17) yields (16). Since this holds for every (g_q, π_q), taking the infimum over both gives the stated bound. □

Remark 4 (Relationship to worst-case bounds). The average cluster radius $\rho_{\mathrm{dec}}^{\mathrm{avg},q}(C)$ and the worst-case cluster radius $\rho_{\mathrm{dec}}^{q}(C)$ are related by

$$\rho_{\mathrm{dec}}^{\mathrm{avg},q}(C) \ \leq\ \rho_{\mathrm{dec}}^{q}(C)$$

for any $C \subseteq \mathcal{X}_q$ and any distribution $\mathcal{D}_q$. Equality holds when $\mathcal{D}_q$ is supported on the worst-case history within each cluster. In general, the gap can be substantial: if a cluster contains one high-gap history with negligible probability, the worst-case radius is large but the average radius remains small. This is precisely the regime in which Proposition 2 provides a tighter characterization than $\epsilon_\infty^\star(K; q)$ alone.

To make this concrete, consider $C = \{h_1, h_2\}$ with $\Pr(Z = h_1) = 1 - \delta$ and $\Pr(Z = h_2) = \delta$ for small $\delta > 0$. Suppose $\rho_{\mathrm{dec}}^{q}(\{h_1\}) = 0$, but $\rho_{\mathrm{dec}}^{q}(\{h_1, h_2\}) = 1$. Then $\rho_{\mathrm{dec}}^{q}(C) = 1$, but $\rho_{\mathrm{dec}}^{\mathrm{avg},q}(C) \leq \delta$: the average-case floor is δ while the worst-case floor is 1.

C DeMem: Full Construction and Analysis

C.1 Confidence bounds and distance certificates

Let $N := |\mathcal{X}|$. For each observed context–action pair (x, a), let $n_t(x, a)$ be the pull count and $\hat{\mu}_t(x, a)$ the empirical mean. For $n_t(x, a) > 0$, define

$$c_t(x, a) := \sqrt{\frac{\log(4NAt^2/\delta)}{2\, n_t(x, a)}}. \tag{19}$$

For $n_t(x, a) = 0$, we set the interval to the trivial bound $[0, 1]$.

Let

$$\mathrm{UCB}_t(x, a) := \begin{cases} \min\{1, \hat{\mu}_t(x, a) + c_t(x, a)\}, & n_t(x, a) > 0, \\ 1, & n_t(x, a) = 0, \end{cases}$$

$$\mathrm{LCB}_t(x, a) := \begin{cases} \max\{0, \hat{\mu}_t(x, a) - c_t(x, a)\}, & n_t(x, a) > 0, \\ 0, & n_t(x, a) = 0. \end{cases}$$

and define

$$\mathrm{UCB}_t^\star(x) := \max_{a\in\mathcal{A}} \mathrm{UCB}_t(x, a), \qquad \mathrm{LCB}_t^\star(x) := \max_{a\in\mathcal{A}} \mathrm{LCB}_t(x, a).$$

These induce gap bounds

$$\overline{\Delta}_t(x, a) := \mathrm{UCB}_t^\star(x) - \mathrm{LCB}_t(x, a), \qquad \underline{\Delta}_t(x, a) := \mathrm{LCB}_t^\star(x) - \mathrm{UCB}_t(x, a).$$

Define corresponding upper and lower certificates for decision distance:

$$\overline{d}_t(x, x') := \min_{a\in\mathcal{A}} \max\{\overline{\Delta}_t(x, a), \overline{\Delta}_t(x', a)\}, \qquad \underline{d}_t(x, x') := \min_{a\in\mathcal{A}} \max\{\underline{\Delta}_t(x, a), \underline{\Delta}_t(x', a)\}. \tag{20}$$

For any nonempty cluster C of observed contexts, define the corresponding cluster-radius certificates

$$\overline{\rho}_t(C) := \min_{a\in\mathcal{A}} \max_{x\in C} \overline{\Delta}_t(x, a), \qquad \underline{\rho}_t(C) := \min_{a\in\mathcal{A}} \max_{x\in C} \underline{\Delta}_t(x, a). \tag{21}$$

Lemma 1 (Validity of the certificates)**.** *With probability at least $1-\delta$, for all observed contexts, all sampled pairs (x, x'), all nonempty observed clusters C, and all times t,*

$$\underline{d}_t(x,x') \le d_{\mathrm{dec}}(x,x') \le \overline{d}_t(x,x'), \qquad \underline{\rho}_t(C) \le \rho_{\mathrm{dec}}(C) \le \overline{\rho}_t(C).$$

Proof. For any fixed $x \in \mathcal{X}$, $a \in \mathcal{A}$, and $t \ge 1$ with $n_t(x,a) > 0$, Hoeffding's inequality gives

$$\Pr(|\hat{\mu}_t(x,a) - \mu(x,a)| > c_t(x,a)) \le 2\exp\bigl(-2n_t(x,a)c_t(x,a)^2\bigr) = \frac{\delta}{2NAt^2}.$$

Taking a union bound over N contexts, A actions, and all $t \ge 1$, the total failure probability is at most

$$\sum_{t=1}^{\infty} NA \cdot \frac{\delta}{2NAt^2} = \frac{\delta}{2}\sum_{t=1}^{\infty}\frac{1}{t^2} < \delta.$$

Therefore, with probability at least $1-\delta$,

$$\mathrm{LCB}_t(x,a) \le \mu(x,a) \le \mathrm{UCB}_t(x,a) \qquad \text{for all } x \in \mathcal{X},\ a \in \mathcal{A}, \text{ and all } t.$$

This implies

$$\underline{\Delta}_t(x,a) \le \Delta(x,a) \le \overline{\Delta}_t(x,a)$$

uniformly. Taking $\min_a \max\{\cdot,\cdot\}$ gives the pairwise distance bounds, and taking $\min_a \max_{x\in C}$ gives the cluster-radius bounds. □

C.2 Frozen protocol for choosing the distortion level

Cannot-link graph. At the beginning of epoch e, with start time $t_e = 2^{e-1}$, define a cannot-link graph $G_\alpha = (V_e, E_\alpha)$ over the observed contexts $V_e = \{X_1, \dots, X_{t_e-1}\}$: for a candidate pairwise level α, add an edge (x, x') whenever

$$\underline{d}_{t_e}(x,x') > \alpha.$$

If a cluster C has true decision radius $\rho_{\mathrm{dec}}(C) \le \alpha$, then every pair in C must satisfy $d_{\mathrm{dec}}(x,x') \le \alpha$, and hence C contains no certified cannot-link edge on the confidence event. Thus graph independence is a necessary condition for small cluster radius.

The converse is not generally true: because d_{dec} need not satisfy the triangle inequality, pairwise compatibility does not imply the existence of a single shared action for the whole cluster. We therefore use the graph only as a conservative splitting constraint. The actual compression price of a returned partition $\mathcal{P}$ is measured by the cluster-level certificate

$$\overline{\epsilon}_{t_e}(\mathcal{P}) \coloneqq \max_{C\in\mathcal{P}} \overline{\rho}_{t_e}(C), \tag{22}$$

which upper-bounds $\max_{C\in\mathcal{P}} \rho_{\mathrm{dec}}(C)$ with high probability.

For the graph relaxation, define

$$\alpha^\star_{\mathrm{graph}}(K) \coloneqq \inf\bigl\{\alpha \ge 0 :\ \chi(G_\alpha) \le K\bigr\}. \tag{23}$$

This is a pairwise graph threshold, not the decision-distortion frontier $\epsilon^\star_\infty(K)$. By Theorem 3, computing $\chi(G_\alpha)$ is NP-hard, so we replace it with a polynomial-time greedy coloring.

Greedy partition subroutine. Algorithm 2 sorts the candidate pairwise graph levels and then binary-searches over them using the monotone feasibility condition

$$\mathrm{degen}(G_\alpha) + 1 \le K.$$

This condition is monotone in α: increasing α removes cannot-link edges and therefore cannot increase graph degeneracy. Once the smallest feasible level α_e is found, the algorithm greedily colors G_{α_e} using a degeneracy ordering. Since degeneracy-order greedy coloring uses at most $\mathrm{degen}(G_{\alpha_e}) + 1 \le K$ colors, the returned partition satisfies the K-slot budget.

The candidate set has size $O(|V_e|^2)$, and sorting it costs $O(|V_e|^2 \log |V_e|)$. Each binary-search step constructs G_α and computes its degeneracy in $O(|V_e|^2)$ time, with $O(\log |V_e|)$ steps. Evaluating the certified cluster radii of the returned partition costs $O(KA|V_e|)$. Thus the per-epoch cost is

$$O\bigl(|V_e|^2 \log |V_e| + KA|V_e|\bigr).$$

Algorithm 2 GREEDY-PARTITION: binary-search epoch partition

Input: pairwise cannot-link certificates $\{\underline{d}_{t_e}(x, x')\}$, cluster-radius certificates $\{\overline{\rho}_{t_e}(C)\}$, budget K

Let V_e be the set of observed contexts at epoch e
Sort candidate graph levels

$$\mathcal{E} \leftarrow \{0\} \cup \{\underline{d}_{t_e}(x, x') : \ x, x' \in V_e, \ x \neq x'\}$$

in increasing order
$\ell \leftarrow 1, \ r \leftarrow |\mathcal{E}|, \ i^\star \leftarrow |\mathcal{E}|$
while $\ell \leq r$ **do**
    $i \leftarrow \lfloor (\ell + r)/2 \rfloor, \alpha \leftarrow \mathcal{E}_i$
    Construct $G_\alpha = (V_e, E_\alpha)$ with edges

$$(x, x') \in E_\alpha \quad \Longleftrightarrow \quad \underline{d}_{t_e}(x, x') > \alpha$$

    Compute the degeneracy $d_\alpha = \mathrm{degen}(G_\alpha)$
    if $d_\alpha + 1 \leq K$ **then**
        $i^\star \leftarrow i$
        $r \leftarrow i - 1$
    else
        $\ell \leftarrow i + 1$
    end if
end while
$\alpha_e \leftarrow \mathcal{E}_{i^\star}$
Construct $G_{\alpha_e} = (V_e, E_{\alpha_e})$
Compute a degeneracy ordering of G_{α_e}
Greedily color G_{α_e} along this ordering, assigning each vertex the smallest-index color not used by its already-colored neighbors
Let $\mathcal{P}_e = \{C_1, \ldots, C_K\}$ be the induced partition, adding empty clusters if fewer than K colors are used
Set

$$\epsilon_e^{\mathrm{cert}} \leftarrow \max_{C \in \mathcal{P}_e} \overline{\rho}_{t_e}(C)$$

return partition $\mathcal{P}_e$, graph level α_e, certified distortion $\epsilon_e^{\mathrm{cert}}$

Structural guarantee. Recall that the *degeneracy* $\mathrm{degen}(G)$ of a graph G is the smallest d such that every subgraph of G has a vertex of degree at most d. Greedy coloring with smallest-last ordering uses at most $\mathrm{degen}(G) + 1$ colors [see, e.g., 23]. This yields the following gap characterization.

Proposition 3 (Greedy graph-relaxation gap)**.** *Let $\alpha^\star = \alpha^\star_{\mathrm{graph}}(K)$ and $G^\star = G_{\alpha^\star}$. Let α_e be the graph level returned by Algorithm 2. Then:*

(i) $\alpha_e \geq \alpha^\star_{\mathrm{graph}}(K)$ *always.*
(ii) $\alpha_e = \alpha^\star_{\mathrm{graph}}(K)$ *whenever* $\mathrm{degen}(G^\star) + 1 \leq K$.
(iii) In general,

$$\alpha_e \ \leq \ \inf\{\alpha \geq 0 : \ \mathrm{degen}(G_\alpha) + 1 \leq K\}. \tag{24}$$

These statements concern the pairwise graph relaxation only. The cluster-level compression price used in Theorem 4 is $\epsilon_e^{\mathrm{cert}}$.

Proof. (i) The greedy coloring can only return at a level α for which it uses at most K colors. This implies that G_α is K-colorable, so $\alpha \geq \alpha^\star_{\mathrm{graph}}(K)$.

(ii) If $\mathrm{degen}(G^\star)+1 \leq K$, then the smallest-last greedy coloring of $G^\star$ uses at most $\mathrm{degen}(G^\star)+1 \leq K$ colors. Hence Algorithm 2 returns at level $\alpha^\star$ or earlier. Combining this with part (i) gives equality.

(iii) For any α satisfying $\mathrm{degen}(G_\alpha) + 1 \leq K$, smallest-last greedy coloring uses at most K colors, so the algorithm must return no later than that α. Taking the infimum over such levels yields the claim. □

Remark 5 (Sparsity in practice). In agent-memory applications, the cannot-link graph is typically sparse: most context pairs do not trigger certified incompatibility. On LoCoMo, only 4.6% of routing events trigger a certified split (Table 9), implying low edge density and hence low degeneracy in the

graph relaxation. Appendix E.15 reports this graph-level behavior. Separately, the regret analysis and empirical distortion measurements use the cluster-level certificate $\epsilon_e^{\mathrm{cert}}$, so they do not rely on pairwise compatibility composing into cluster compatibility.

C.3 Algorithm

C.4 Certification exploration

Quantities used in Theorem 4. For the partition $\mathcal{P}_e$ used in epoch e, define

$$\epsilon_e^{\mathrm{cert}} \coloneqq \max_{C \in \mathcal{P}_e} \overline{\rho}_{t_e}(C), \qquad \bar{\epsilon}_T^{\mathrm{cert}} \coloneqq \max_{e \le \lceil \log_2 T \rceil} \epsilon_e^{\mathrm{cert}}.$$

The terminal certification threshold is

$$B_T(\gamma) \coloneqq \left\lceil \frac{8 \log(4NAT^2/\delta)}{\gamma^2} \right\rceil.$$

The cannot-link certificates in Section 4.1 are computed from context-level statistics, whereas exploitation is performed using cluster-level UCB statistics. Without explicit context-level exploration, a context routed to an already confident cluster may stop receiving enough action-level samples for its own confidence intervals to shrink. In that case, a genuine decision conflict inside the cluster could remain uncertified.

For this reason, DeMem samples each observed context–action pair until

$$n_t(x, a) \ge B_t(\gamma) \coloneqq \left\lceil \frac{8 \log(4NAt^2/\delta)}{\gamma^2} \right\rceil.$$

After this coverage condition holds, the context-level confidence radius is $O(\gamma)$, and the algorithm returns to cluster-level UCB. Thus pairwise conflicts separated by a positive margin larger than γ can be certified once the relevant contexts have been observed often enough. Zero-margin conflicts, or conflicts involving rarely observed contexts, are not forced to be certified; their effect is instead captured by the realized cluster distortion term in Theorem 4.

C.5 Proof sketch of the upper bound

Proof sketch of Theorem 4. We decompose regret into three parts.

Realized compression price. At each epoch e, the frozen partition $\mathcal{P}_e$ induces a K-state encoder $g^{(e)}$. On the confidence event of Lemma 1, its true cluster distortion is bounded by

$$\max_{C \in \mathcal{P}_e} \rho_{\mathrm{dec}}(C) \le \max_{C \in \mathcal{P}_e} \overline{\rho}_{t_e}(C) = \epsilon_e^{\mathrm{cert}}.$$

Comparing against the best action available under $g^{(e)}$ in each epoch, the compression cost over the full horizon is at most

$$\sum_e T_e \, \epsilon_e^{\mathrm{cert}} \le T \, \bar{\epsilon}_T^{\mathrm{cert}}.$$

Statistical learning within epochs. Condition on an epoch in which the encoder is frozen and ignore rounds used for certification exploration. On the remaining rounds, the learner faces a standard bandit problem over K memory states and A actions. A standard UCB analysis gives $\tilde{O}(\sqrt{AKT_e})$ regret over an epoch of length T_e, and summing over epochs yields $\tilde{O}(\sqrt{AKT})$.

Context-level certification exploration. The certification branch ensures that, for every observed context x and action a, the algorithm samples (x, a) until $n_t(x, a) \ge B_t(\gamma)$. Hence each distinct context contributes at most $AB_T(\gamma)$ such exploration rounds over the horizon, and the total regret from certification exploration is at most

$$O\big(AN_T B_T(\gamma)\big) = \tilde{O}(AN_T/\gamma^2).$$

After this coverage condition holds, the context-level confidence radius is at most $O(\gamma)$. Therefore any pairwise conflict separated by margin larger than γ can be certified once the corresponding contexts have both reached coverage. Conflicts with vanishing margin are not guaranteed to be certified, but they are charged through the realized cluster distortion term rather than through an invalid pairwise-to-cluster implication.

Combining the three terms proves the stated bound. □

Algorithm 3 DeMem: Decision-aware Memory with Certified Splits

For contexts observed before epoch e, the encoder $g^{(e)}$ is the cluster lookup induced by $\mathcal{P}_e$. For a context first observed inside the epoch, DeMem extends $g^{(e)}$ using a deterministic fallback router ϕ_e, which assigns the context to the cluster that minimizes the certified radius after insertion. This temporary assignment is kept fixed until the next epoch rebuilds the cannot-link graph and the K-slot partition.

Input: memory budget K, confidence δ, certification resolution $\gamma > 0$

Initialize an empty context table and context-level statistics $(n(x,a), \hat{\mu}(x,a))$
Initialize a single cluster $\mathcal{C}^{(1)} = \{C_1\}$ with $C_1 \leftarrow \emptyset$
for epoch $e = 1, 2, \ldots$ **do**
    $t_e \leftarrow 2^{e-1}$
    Using data up to $t_e - 1$, compute $\underline{d}_{t_e}(x, x')$ for all observed pairs and $\overline{\rho}_{t_e}(C)$ for candidate clusters
    $(\mathcal{P}_e,\ \alpha_e,\ \epsilon_e^{\mathrm{cert}}) \leftarrow$ GREEDY-PARTITION$\big(\{\underline{d}_{t_e}(x,x')\}, \{\overline{\rho}_{t_e}(C)\}, K\big)$ ▷ Algorithm 2
    Construct epoch clusters $C_1^{(e)}, \ldots, C_K^{(e)}$ from $\mathcal{P}_e$, and define $g^{(e)}(x)$ as the cluster index for every previously observed context x
    Define a deterministic fallback router for out-of-sample contexts:

$$\phi_e(x) \in \arg\min_{j \in [K]} \overline{\rho}_t\Big(C_j^{(e)} \cup \{x\}\Big),$$

    with deterministic tie-breaking
    Reset cluster-level statistics $(n_{m,a}, \hat{\mu}_{m,a})$
    for $t = t_e, \ldots, 2^e - 1$ **do**
        Observe context $x \leftarrow X_t$
        if x is new **then**
            Add x to the context table and initialize $(n(x,a), \hat{\mu}(x,a))$
            $m \leftarrow \phi_e(x)$
            Add x to $C_m^{(e)}$ and set $g^{(e)}(x) \leftarrow m$
        end if
        Let $m \leftarrow g^{(e)}(x)$
        Set

$$B_t(\gamma) \coloneqq \left\lceil \frac{8 \log(4NAt^2/\delta)}{\gamma^2} \right\rceil.$$

        if there exists $a \in \mathcal{A}$ with $n(x,a) < B_t(\gamma)$ **then**
            Play a certification-exploration action

$$A_t \in \arg\min_{a \in \mathcal{A}} n(x,a).$$

        else
            Play the cluster-level UCB action

$$A_t \in \arg\max_{a \in \mathcal{A}} \left(\hat{\mu}_{m,a} + \sqrt{\frac{2 \log(4AKt^2/\delta)}{\max\{1, n_{m,a}\}}} \right).$$

        end if
        Observe reward R_t
        Update cluster-level statistics $(n_{m,A_t}, \hat{\mu}_{m,A_t})$
        Update context-level statistics $(n(x,A_t), \hat{\mu}(x,A_t))$
    end for
end for

Remark 6 (Optimality up to realized compression). Theorem 5 shows that any K-state algorithm incurs

$$\Omega\Big(T\epsilon^\star_\infty(K) + \sqrt{AKT}\Big).$$

Theorem 4 matches the minimax statistical learning term up to logarithmic factors. The remaining difference is the excess realized compression price

$$\bar{\epsilon}^{\mathrm{cert}}_T - \epsilon^\star_\infty(K),$$

together with the context-certification cost $\tilde{O}(AN_T/\gamma^2)$. Thus DeMem is near-minimax whenever the returned partitions have certified cluster distortion close to the optimal decision-distortion frontier and the number of newly observed contexts is not the dominant term.

C.6 Proof sketch of the lower bound

Proof sketch of Theorem 5. The lower bound has two orthogonal sources.

Compression term. The runtime memory constraint alone implies that no deterministic K-state answer-time policy can beat $\epsilon^\star_\infty(K)$ in worst-case distortion. Therefore the regret must contain a term of order $T\epsilon^\star_\infty(K)$.

Statistical term. Consider an instance with $\epsilon^\star_\infty(K) = 0$ in which $\mathcal{X} = [K]$ and each context induces an independent A-armed bandit subproblem. Each subproblem appears T/K times in expectation, so the classical minimax lower bound gives $\Omega(\sqrt{A(T/K)})$ regret per state. Summing over the K states yields $\Omega(\sqrt{AKT})$.

A direct-sum construction combines the two terms, giving

$$\Omega\Big(T\epsilon^\star_\infty(K) + \sqrt{AKT}\Big).$$

□

C.7 Extension to non-stationary rewards

The regret guarantees of Section 4.2 assume a fixed reward function μ. In many agent settings the decision-relevant structure may shift over time—for example, when user preferences evolve or the task distribution changes across sessions. This appendix states and proves the extension to a standard block-stationary model, together with a matching lower bound.

The block-stationary setting captures the simplest version of this phenomenon: within each stationary block the theory of Sections 3.4–4 applies unchanged. When restart boundaries align with the stationary segments, the additional statistical cost of non-stationarity is the standard $\sqrt{S+1}$ factor. For fixed restarts with arbitrary change-point locations, the wrapper may mix data from two regimes inside a restart interval; this contributes an additional $O(SL)$ misalignment term, where L is the restart period. Extending to smoothly drifting, adversarial, or adaptively detected changes is an interesting direction for future work.

Setting. The reward function is piecewise constant: there exist change points $1 = \tau_1 < \tau_2 < \cdots < \tau_{S+1} \le T$ such that within each block $[\tau_s, \tau_{s+1})$ the reward function $\mu_s : \mathcal{X} \times \mathcal{A} \to [0,1]$ is fixed. Let $T_s := \tau_{s+1} - \tau_s$ denote the length of block s, so that $\sum_{s=1}^{S+1} T_s = T$. For each block s, let $\epsilon^\star_{\infty,s}(K)$ denote the worst-case K-state distortion under μ_s, and define

$$\epsilon^\star_{\infty,\max}(K) := \max_{s\in[S+1]} \epsilon^\star_{\infty,s}(K).$$

Algorithm: restarting DeMem. We consider a restart wrapper with restart interval length L. Let $M := \lceil T/L \rceil$ be the number of restart intervals. At the start of each interval, the wrapper resets DeMem: it clears all context-level and cluster-level statistics, reinitializes the partition to a single cluster, and restarts the doubling-epoch schedule of Algorithm 3.

If the restart boundaries align with, or refine, the true stationary segments, each restart interval lies inside a single reward regime. If a true change point falls inside a restart interval, that interval may

mix pre- and post-change data. Since per-round regret is at most one, each such contaminated interval contributes at most L additional regret. Thus arbitrary misaligned change points introduce an additive $O(SL)$ term.

Proposition 4 (Non-stationary regret with fixed restarts)**.** *Suppose the reward function changes at most S times over T rounds. Run the restarting DeMem wrapper with fixed restart interval length L, and let $M \coloneqq \lceil T/L \rceil$. With probability at least $1-\delta$,*

$$\mathrm{Reg}(T) \;\le\; T \cdot O\big(\epsilon^{\star}_{\infty,\max}(K)\big) \;+\; \tilde{O}\big(\sqrt{AKTM}\big) \;+\; O\big(AMN_T B_T(\gamma)\big) \;+\; O(SL). \tag{25}$$

The final term $O(SL)$ accounts for restart intervals that contain misaligned change points. If the restart schedule aligns with or refines the true stationary segments, this term is zero.

Proof. Partition the horizon into M restart intervals $I_1, \ldots, I_M$, each of length at most L. Call an interval *contaminated* if it contains a true change point. There are at most S contaminated intervals. Since one-step regret is bounded by one, their total contribution is at most $O(SL)$.

On every non-contaminated interval I_j, the reward function is fixed. Applying Theorem 4 to interval I_j, with confidence parameter δ/M, gives

$$\mathrm{Reg}_j \le |I_j| \cdot O\big(\epsilon^{\star}_{\infty,j}(K)\big) + \tilde{O}\big(\sqrt{AK|I_j|}\big) + O\big(AN_{I_j} B_T(\gamma)\big),$$

where N_{I_j} is the number of distinct contexts observed in interval I_j. Summing over non-contaminated intervals and absorbing logarithmic factors from the union bound gives the three main terms.

For the compression term,

$$\sum_j |I_j| \epsilon^{\star}_{\infty,j}(K) \le T \epsilon^{\star}_{\infty,\max}(K).$$

For the statistical term, Cauchy–Schwarz gives

$$\sum_{j=1}^{M} \sqrt{AK|I_j|} \le \sqrt{M \cdot AK \sum_{j=1}^{M} |I_j|} = \sqrt{AKTM}.$$

For certification exploration, a context can be rediscovered after each restart, so

$$\sum_{j=1}^{M} N_{I_j} \le M N_T,$$

which yields $O(AMN_T B_T(\gamma))$. Adding the contaminated-interval cost $O(SL)$ proves (25). □

Remark 7 (Adaptive restarts). When S is unknown or change points are not aligned with a fixed restart schedule, one can replace fixed restarts with a change-detection mechanism that monitors shifts in reward estimates or cannot-link certificate rates. In such a scheme, the $O(SL)$ term would be replaced by a detection-delay cost depending on the separation between pre- and post-change reward functions. Providing sharp adaptive guarantees for DeMem requires a separate change-detection analysis, which we leave to future work.

D Agent Memory Instantiation and Theory–Implementation Alignment

D.1 Why an instantiation is needed

The theory models memory as a finite runtime state $M_t \in [K]$ induced by an encoder $g : \mathcal{X} \to [K]$. To realize this abstraction in an agent-memory system, we instantiate a minimal K-slot mechanism in which histories are routed into a bounded set of reusable slot identities at answer time. This slot-based view is the operational form of the budgeted memory model studied in the main text. The remaining implementation choices—including slot realizations, update rules, and routing scores—specify how these budgeted decision states are represented and maintained within an agent pipeline.

D.2 System components

At each step, an agent receives a context x_t and must make a downstream decision under a strict runtime budget of K discrete states. The implementation contains three modules:

1. **Memory slots** $\{\mathcal{S}_k\}_{k=1}^{K}$ storing compressed state information.
2. **Router** $e_\theta(x_t) \in [K]$ selecting a slot identity.
3. **Policy** π_ϕ, whose downstream behavior is conditioned on the selected slot and its maintained state realization.

Memory control versus downstream decision. It is important to separate memory control from task decision. Routing, slot reading, slot updating, and certified splitting are memory operations that implement the encoder and state-maintenance mechanism. By contrast, the action in the abstract model corresponds to the downstream decision supported by the selected slot. In a language agent, this downstream decision may be realized as an answer, a tool choice, or another task-level control output conditioned on the selected memory state.

Conversational-benchmark realization. In the LoCoMo and LongMemEval experiments, these components are instantiated as follows.

- **Context** $x_t = (h_t, q_t)$: the dialogue history h_t accumulated across sessions up to time t, together with the evaluation query q_t.
- **Router** e_θ: assigns (h_t, q_t) to one of K active slots by computing $m_t = \arg\max_{k\in[K]} \mathrm{Score}_\theta(x_t, \mathcal{S}_k)$, where Score_θ is an embedding-similarity score between the query and each slot's current content.
- **Slot content** $\mathcal{S}_{m_t}$: a textual summary of at most L characters, maintained via the update rule in (28). This is the bounded realization of the abstract memory state m_t.
- **Action / Answer**: the answering LLM receives the slot content $\mathcal{S}_{m_t}$ as the sole memory evidence and generates an answer a_t under deterministic decoding. Under this protocol, the action is fully determined by the selected slot identity and its content, so the effective policy $\pi(\cdot \mid m_t)$ is a point mass.
- **Reward** $\mu(x_t, a_t)$: the binary correctness score (LoCoMo) or graded accuracy (LongMemEval) assigned by the held-out LLM judges.

Under this mapping, the abstract model's prediction is concrete: answer quality is bounded by (i) how well the K-slot partition preserves query-relevant distinctions (the compression term $\epsilon^\star_\infty(K)$), (ii) how accurately the router selects the correct slot (the routing term η_{route}), and (iii) how faithfully the bounded slot content supports the correct answer (the realization term η_{read}).Figure 1 visualizes this chain end to end.

D.3 Slot representations

Each slot may store either (i) a bounded textual summary σ_k, or (ii) a compact vector prototype $p_k \in \mathbb{R}^d$, or both. For textual slots, we impose a per-slot token budget L so that

$$|\sigma_k| \le L \qquad \text{for all } k,$$

which keeps the total memory footprint bounded by $K \cdot L$.

D.4 Routing and updates

The router assigns contexts to slots by

$$m_t = \arg\max_{k\in[K]} \mathrm{Score}_\theta(x_t, \mathcal{S}_k), \tag{26}$$

where Score_θ may be implemented by embedding similarity, a small MLP, or an LLM-based routing prompt, as long as the output is discrete.

For vector slots, one may use the update

$$p_{m_t} \leftarrow (1-\eta)p_{m_t} + \eta f(x_t), \tag{27}$$

while textual slots are updated by first extracting an implementation-level memory item

$$u_t \coloneqq \psi_\theta(x_t),$$

and then applying

$$\sigma_{m_t} \leftarrow \mathrm{Compress}_L\big(\sigma_{m_t} \oplus u_t\big), \tag{28}$$

where ψ_θ denotes the textual memory extractor, $\oplus$ denotes concatenation, and $\mathrm{Compress}_L(\cdot)$ enforces the per-slot length limit.

D.5 Certified split as a memory operation

The operational analogue of the theory is a cannot-link constraint. Whenever two contexts routed to the same slot are certified to prefer conflicting actions, the slot is marked inconsistent and becomes eligible for splitting.

A generic empirical trigger is

$$\exists a \neq a' \in \mathcal{A}: \quad \widehat{r}(x,a) - \widehat{r}(x,a') > \beta_t \quad \text{and} \quad \widehat{r}(x',a') - \widehat{r}(x',a) > \beta_t. \tag{29}$$

Intuitively, x prefers a while x' prefers a' with sufficient confidence, so merging them induces nontrivial decision distortion.

D.6 Executing the split

Given a cannot-link witness pair (x, x') inside an active slot k, DeMem creates a new slot k'. To avoid circular dependence between repartitioning and slot contents, the split is first computed using temporary seed contents derived from the witness pair, rather than from the yet-uninitialized post-split slots.

Let

$$\mathcal{A}_k^{(0)} \coloneqq \mathrm{MemInit}(\{x\}), \qquad \mathcal{A}_{k'}^{(0)} \coloneqq \mathrm{MemInit}(\{x'\})$$

be two temporary anchors. For textual memory, $\mathrm{MemInit}(\{x\})$ denotes the bounded summary or compressed evidence item generated from x; for vector memory, it denotes the prototype $f(x)$. These anchors are used only to compute the split assignment. For every item z currently assigned to slot k, define

$$e_\theta(z) = \begin{cases} k, & \mathrm{Score}_\theta(z, \mathcal{A}_k^{(0)}) \geq \mathrm{Score}_\theta(z, \mathcal{A}_{k'}^{(0)}), \\ k', & \text{otherwise}, \end{cases} \tag{30}$$

with deterministic tie-breaking.

Let

$$P_k = \{z : e_\theta(z) = k\}, \qquad P_{k'} = \{z : e_\theta(z) = k'\}.$$

The actual slot contents are then reinitialized from the resulting partitions:

$$\mathcal{S}_k \leftarrow \mathrm{MemInit}(P_k), \qquad \mathcal{S}_{k'} \leftarrow \mathrm{MemInit}(P_{k'}).$$

Thus routing scores in Eq. (30) depend only on fixed temporary anchors derived from the witness pair, while the final slot contents are computed after the partition is determined.

D.7 Runtime loop

Capacity handling. In all reported experiments, DeMem handles the strict K-slot capacity by threshold-based split suppression rather than by merge or eviction. When free slots remain, split witnesses are tested with the default threshold τ_{split}. Once all K slots are active, the system switches to the more conservative threshold $\tau_{\mathrm{sat}} \geq \tau_{\mathrm{split}}$; witnesses that still exceed this threshold are logged in $\mathcal{C}_{\mathrm{sat}}$, but the corresponding split is not executed. No LRU eviction, semantic merging, or slot replacement is used in the reported experiments.

This policy preserves the hard K-slot budget without introducing an additional memory-management heuristic. Conflicts discovered after capacity saturation may therefore remain unresolved and are reflected in the realized compression or routing error.

Algorithm 4 Budgeted Agent Memory with Certified Splits

Input: Budget K, per-slot token limit L, confidence schedule β_t, default split threshold τ_{split}, saturated split threshold $\tau_{\text{sat}} \geq \tau_{\text{split}}$

Initialize active slots $\{\mathcal{S}_k\}_{k=1}^{K_0}$ with $K_0 \leq K$
Initialize cannot-link sets $\mathcal{C} \leftarrow \emptyset$ and $\mathcal{C}_{\text{sat}} \leftarrow \emptyset$
for $t = 1, 2, \ldots$ **do**
    Observe context x_t
    Route:

$$m_t \leftarrow \arg \max_{k:\mathcal{S}_k \text{ active}} \text{Score}_\theta(x_t, \mathcal{S}_k)$$

    Read slot content

$$\tilde{s}_t \leftarrow \text{read}(\mathcal{S}_{m_t})$$

    Act:

$$a_t \sim \pi_\phi(\cdot \mid m_t, \tilde{s}_t)$$

    Receive reward r_t
    Update reward statistics
    Update slot content via (27) or (28)
    Set the active split threshold

$$\tau_t \leftarrow \begin{cases} \tau_{\text{split}}, & \text{if the number of active slots is } < K, \\ \tau_{\text{sat}}, & \text{otherwise,} \end{cases} \qquad \tau_{\text{sat}} \geq \tau_{\text{split}}.$$

    Check certified inconsistency via (29) using threshold τ_t
    if a cannot-link witness is found **then**
        if number of active slots $< K$ **then**
            Add the witness to $\mathcal{C}$
            Split the slot using (30)
        else
            Add the witness to $\mathcal{C}_{\text{sat}}$
            Suppress the split; no slot is created, merged, or evicted ▷ capacity-saturated thresholding
        end if
    end if
end for

D.8 Alignment with the theory

The correspondence between the abstract model and the implementation is direct:

Theory	**Generic Implementation**	**LoCoMo / LongMemEval**
context $x = (h, q)$	history plus current query	multi-session dialogue history h and evaluation query q
query q	decision-time side information bypassing the memory bottleneck	user/evaluation question given directly to the LLM
encoder $g : \mathcal{H} \times \mathcal{Q} \to [K]$	query-aware router e_θ	$m = \arg\max_k \text{Score}_\theta((h, q), \mathcal{S}_k)$
memory state $m \in [K]$	selected slot identity	selected slot identity + bounded textual summary ($\leq L$ chars)
policy $\pi(\cdot \mid m, q)$	slot- and query-conditioned downstream decision rule	deterministic LLM answer conditioned on selected slot content and query
action $a \in \mathcal{A}$	generated downstream decision	generated answer string
reward $\mu(x, a)$	task reward	binary correctness (LoCoMo) / graded accuracy (LongMemEval)
cannot-link certificate	guarded split trigger	candidate-restricted answer conflict between co-routed histories for a query
hard budget K	number of active slots	number of active slots
distortion ϵ	conflict tolerance / split threshold	answer-quality loss from merging decision-incompatible histories

The correspondence is at the level of runtime decision states. The query is provided directly to the policy at answer time, while the dialogue history is accessed through one of K selected slot identities. Thus the memory budget limits the number of runtime states available for representing history, not the number of possible natural-language answers across all queries. In the language-agent implementation, each slot additionally maintains a bounded realization—for example, a textual summary, a compact prototype, or both—through which the selected memory state is materialized for answer generation. These realizations do not change the underlying budgeted state abstraction; they specify how each discrete memory state is represented inside the agent pipeline.

Reading the table. Each row shows a single concept at three levels of abstraction. The leftmost column states the formal object from Sections 3–3.2; the middle column gives the generic system component from Algorithm 4; the rightmost column specifies the exact realization used in the conversational benchmarks of Section 5. This three-column view makes it possible to trace any theoretical statement, such as the forgetting boundary of Theorem 1 or the regret decomposition of Theorem 4, to its operational counterpart in the experimental system. A concrete instance of this trace is shown in Figure 1.

This decomposition also implies that the routing criterion is modular: because η_{route} enters additively and independently of the compression floor and the realization term, replacing the retrieval or selection stage of an existing memory architecture with decision-aware routing can reduce distortion without modifying the remaining pipeline. We validate this composability empirically in Appendix E.12: applying the decision-aware selection criterion to two existing memory systems without modifying their storage or generation layers improves both on LoCoMo.

D.9 A formal approximation bridge from abstract states to slot-based agents

Beyond the conceptual alignment, the slot-based view yields a simple approximation guarantee. The key point is that the abstract theory controls the *dominant compression term*, while the concrete implementation contributes only additional approximation errors due to routing and bounded state realization.

Let $(g^\star, \{a_m^\star\}_{m=1}^K)$ be any K-state decision rule, where

$$g^\star : \mathcal{X} \to [K], \qquad a_m^\star \in \mathcal{A}.$$

Its worst-case distortion is

$$\sup_{x \in \mathcal{X}} \big(\mu^\star(x) - \mu(x, a^\star_{g^\star(x)})\big).$$

In particular, one may take $(g^\star, \{a_m^\star\})$ to be an optimal solution achieving $\epsilon^\star_\infty(K)$.

Now consider a slot-based implementation with router e_θ, slot memory $\{\mathcal{S}_k\}_{k=1}^K$, slot readout $\mathrm{read}(\mathcal{S}_k)$, and downstream policy π_ϕ.

We separate the implementation gap into two terms.

Routing error. Assume that under $X \sim D$,

$$\Pr\big[e_\theta(X) \neq g^\star(X)\big] \leq \eta_{\mathrm{route}}. \tag{31}$$

State-realization error. Assume that whenever the router selects the same state as the abstract encoder, the implemented slot-conditioned policy loses at most η_{read} additional reward relative to the abstract slot action:

$$\mathbb{E}\Big[\mu\Big(X, a^\star_{g^\star(X)}\Big) - \mu\big(X, A^{\mathrm{impl}}\big) \,\Big|\, e_\theta(X) = g^\star(X)\Big] \leq \eta_{\mathrm{read}}, \tag{32}$$

where $A^{\mathrm{impl}} \sim \pi_\phi\big(\cdot \mid e_\theta(X), \mathrm{read}(\mathcal{S}_{e_\theta(X)})\big)$.

Proposition 5 (Approximation bridge for slot-based memory)**.** *Under* (31)–(32)*, the average distortion of the implemented slot-based agent satisfies*

$$\mathbb{E}_{X\sim D}\big[\mu^\star(X) - \mu\big(X, A^{\mathrm{impl}}\big)\big] \leq \epsilon^\star_\infty(K) + \eta_{\mathrm{route}} + \eta_{\mathrm{read}}. \tag{33}$$

More generally, for any comparator $(g^\star, \{a_m^\star\})$ with worst-case distortion at most $\bar\epsilon$, the same argument yields

$$\mathbb{E}\big[\mu^\star(X) - \mu\big(X, A^{\mathrm{impl}}\big)\big] \leq \bar\epsilon + \eta_{\mathrm{route}} + \eta_{\mathrm{read}}.$$

Method	Overall Score
DEMEM-CORE	**90.8**

Table 2: Overall LoCoMo result for the reduced DEMEM-CORE variant under the same evaluation protocol as the main experiments.

Proof. Decompose the loss according to whether the learned router agrees with the abstract encoder:

$$\mu^\star(X) - \mu(X, A^{\mathrm{impl}}) = \underbrace{\mu^\star(X) - \mu\Big(X, a^\star_{g^\star(X)}\Big)}_{\text{compression term}} + \underbrace{\mu\Big(X, a^\star_{g^\star(X)}\Big) - \mu(X, A^{\mathrm{impl}})}_{\text{implementation term}}.$$

The first term is at most $\epsilon^\star_\infty(K)$ pointwise by definition of the optimal K-state abstraction. For the second term, condition on the event $\{e_\theta(X) = g^\star(X)\}$. On this event, Assumption (32) bounds the excess loss by η_{read} in expectation. On the complement event, rewards lie in $[0, 1]$, so the additional loss is at most 1. Therefore

$$\mathbb{E}\Big[\mu\Big(X, a^\star_{g^\star(X)}\Big) - \mu(X, A^{\mathrm{impl}})\Big] \le \eta_{\mathrm{read}} + \Pr[e_\theta(X) \neq g^\star(X)] \le \eta_{\mathrm{read}} + \eta_{\mathrm{route}}.$$

Combining the two parts proves (33). □

Interpretation. Proposition 5 identifies how the abstract memory frontier and the realized slot-based system compose. The term $\epsilon^\star_\infty(K)$ captures the irreducible price of compressing histories into K reusable decision states, while routing and bounded slot realization contribute additional approximation terms through η_{route} and η_{read}. This decomposition makes the role of implementation explicit: better routing, more faithful slot realizations, and earlier conflict detection improve the realized agent by reducing the approximation terms, while the underlying memory-compression limit remains governed by the K-state decision abstraction.

D.10 Theory-faithful Core Variant on LoCoMo

To further isolate the contribution of the slot-level memory organization, we evaluate a reduced DeMem variant on **LoCoMo**, denoted DEMEM-CORE. This variant keeps the same budgeted slot structure, routing mechanism, and conflict-driven refinement rule as the full method, while removing the richer generation-time enhancements used in the complete implementation. In this sense, DEMEM-CORE measures how far the slot organization itself can go under the same overall memory regime, before adding the full set of realization and generation components.

Setup. We use the same LoCoMo preprocessing, prompting template, answering backbone, deterministic decoding protocol, and judge setup as in the main experiments. The answer-time memory budget is matched to the main setting. Relative to the full system, DEMEM-CORE uses the same slot budget and the same slot assignment / certified refinement procedure, but relies on a reduced slot realization at answer time rather than the richer memory realization used by the complete model. This yields a more stripped-down version of the method while preserving the same budgeted memory organization.

Result. Under this protocol, DEMEM-CORE reaches an overall LoCoMo score of **90.8**. For compactness, we report the overall score here; the category-wise pattern follows the same qualitative trend as in the main results.

Discussion. The result shows that a substantial portion of the final gain already arises from the budgeted slot organization and the conflict-driven refinement mechanism themselves. The full implementation improves further by using richer slot realizations and a stronger generation-time use of the selected memory state, but the reduced variant is already competitive on its own. This supports the view that the central benefit of DeMem lies in how memory is organized under a fixed budget, rather than only in the additional realization details of the full system.

D.11 Practical split trigger in LLM benchmarks

The certified split rule in Section 4.1 is defined for the online decision model, where repeated context–action feedback supports one-sided certificates of decision incompatibility. One-pass conversational

benchmarks such as LoCoMo provide an answer-level evaluation interface under deterministic decoding. We therefore instantiate the same one-sided refinement principle using candidate-restricted answer-conflict scoring: a split is introduced only when the scored candidate set indicates that no shared candidate memory state performs well for both contexts.

For a context $x = (h, q)$ and a candidate memory slot u, let

$$a(x, u) \coloneqq \mathrm{Ans}_\theta(q, S_u)$$

be the deterministic answer produced from query q and slot contents S_u. Let r_x denote the reference answer for x. The implementation evaluates the answer using B feedback scores. In our main experiments $B = 2$, corresponding to two held-out feedback scorers:

$$s_b(x, u) \coloneqq J_{\mathrm{fb}}^{(b)}\big(a(x, u), r_x\big) \in [0, 1], \qquad b \in [B].$$

We define the answer-level reward estimate and its score dispersion as

$$\widehat{\mu}_{\mathrm{impl}}(x, u) \coloneqq \frac{1}{B}\sum_{b=1}^{B} s_b(x, u), \qquad \widehat{\sigma}^2_{\mathrm{impl}}(x, u) \coloneqq \frac{1}{B-1}\sum_{b=1}^{B}\Big(s_b(x, u) - \widehat{\mu}_{\mathrm{impl}}(x, u)\Big)^2.$$

The dispersion term is used only as a conservative guard band. Specifically, we set

$$\beta_{\mathrm{impl}}(x, u) \coloneqq c_\beta \sqrt{\frac{\widehat{\sigma}^2_{\mathrm{impl}}(x, u) + \sigma_0^2}{B}} + \eta_{\mathrm{cal}},$$

where c_β is a fixed safety multiplier, $\sigma_0 > 0$ is a small floor preventing overconfidence from a small number of scorers, and η_{cal} is a calibration slack chosen on held-out validation instances. This quantity serves as an implementation-level guard band for benchmark scoring, accounting for scorer disagreement and single-pass evaluation noise.

Given a pair of contexts x and x' currently assigned to the same memory block, we construct a small candidate set

$$\mathcal{U}(x, x')$$

of slot-induced decisions. This set includes the current shared slot, the top router-proposed slots for x and x', and, when available, the candidate split slots obtained by materializing separate slot summaries for the two contexts. All counterfactual scores are computed by re-running the deterministic answerer with the corresponding candidate slot and scoring the resulting answer with the same feedback scorers.

For each $u \in \mathcal{U}(x, x')$, because the feedback scores lie in $[0, 1]$, we define clipped lower and upper score envelopes:

$$\mathrm{LCB}_{\mathrm{impl}}(x, u) \coloneqq \max\{0,\, \widehat{\mu}_{\mathrm{impl}}(x, u) - \beta_{\mathrm{impl}}(x, u)\}\,,$$
$$\mathrm{UCB}_{\mathrm{impl}}(x, u) \coloneqq \min\{1,\, \widehat{\mu}_{\mathrm{impl}}(x, u) + \beta_{\mathrm{impl}}(x, u)\}\,.$$

The candidate-restricted analogue of the optimal value for x is

$$\underline{\mu}^\star_{\mathcal{U}}(x) \coloneqq \max_{v \in \mathcal{U}(x, x')} \mathrm{LCB}_{\mathrm{impl}}(x, v).$$

For a shared candidate slot u, we then define a conservative lower estimate of its candidate-restricted decision loss:

$$\widehat{\Delta}^-_{\mathcal{U}}(x, u) \coloneqq \Big[\underline{\mu}^\star_{\mathcal{U}}(x) - \mathrm{UCB}_{\mathrm{impl}}(x, u)\Big]_+.$$

This follows the form of the theoretical suboptimality gap $\Delta(x, a) = \mu^\star(x) - \mu(x, a)$, specialized to the candidate set of slot-induced answers and evaluated with the implementation-level guard band.

We declare a pairwise answer conflict when every shared candidate slot in $\mathcal{U}(x, x')$ incurs a sufficiently large lower-bounded loss for at least one of the two contexts. Formally, define the candidate-restricted radius certificate

$$\widehat{\rho}^-_{\mathrm{impl}}(x, x') \coloneqq \min_{u \in \mathcal{U}(x, x')} \max\Big\{\widehat{\Delta}^-_{\mathcal{U}}(x, u), \widehat{\Delta}^-_{\mathcal{U}}(x', u)\Big\}.$$

A cannot-link edge is added when

$$\widehat{\rho}^-_{\mathrm{impl}}(x, x') > \epsilon_{\mathrm{split}}. \tag{34}$$

The resulting cannot-link graph is then passed to the same budgeted partitioning step used by the memory encoder. Thus the split trigger identifies decision conflicts, while the fixed K-slot budget still controls which conflicts can be separated at runtime.

This rule is one-sided by design. A fired edge indicates candidate-level evidence of decision conflict, while the absence of an edge is not used as a positive compatibility claim. This conservative convention preserves the direction of the theoretical cannot-link logic under deterministic benchmark evaluation.

E Additional Experimental Details

E.1 Synthetic Diagnostic Environment

We construct a controlled *Decoupled Bandit* environment to isolate the effect of memory organization under a fixed runtime memory budget.

Contexts, descriptions, and decision identities. At each round $t = 1, \dots, T$, the environment emits a context

$$c_t = (x_t, z_t),$$

where $x_t \in \mathbb{R}^d$ is a *descriptive representation* and $z_t \in \{1, \dots, M\}$ is a latent *decision identity*. The key property of the environment is that x_t and z_t are only partially aligned: descriptively similar contexts need not share the same optimal action.

Each decision identity z is associated with an action-value vector

$$\mu_z \in [0, 1]^A,$$

where A is the number of available actions. The optimal action for a context c_t is

$$a^\star(c_t) = \arg\max_{a \in [A]} \mu_{z_t, a}.$$

Reward model. After choosing action a_t, the learner receives reward

$$r_t = \mu_{z_t, a_t} + \xi_t,$$

where ξ_t is zero-mean observation noise. The cumulative regret up to round T is

$$\mathrm{Regret}(T) = \sum_{t=1}^{T} \big(\mu_{z_t, a^\star(c_t)} - \mu_{z_t, a_t}\big).$$

Runtime memory budget. At any time, the learner is allowed to maintain at most K reusable memory states (or slots),

$$|\mathcal{M}_t| \le K.$$

Each method defines its own assignment rule from a context c_t to a memory state $m(c_t) \in \mathcal{M}_t$, but all methods are evaluated under the same budget K.

Decision distortion induced by compression. Given a memory assignment $m(c)$, define the induced decision profile of a memory state $s \in \mathcal{M}$ as the average reward vector over the contexts mapped to it,

$$\bar{\mu}_s = \mathbb{E}[\mu_z \mid m(c) = s].$$

The action selected from the compressed memory is

$$\hat{a}(c) = \arg\max_{a \in [A]} \bar{\mu}_{m(c), a}.$$

We define the decision distortion of a memory system as

$$D(\mathcal{M}) = \mathbb{P}\big[\hat{a}(c) \neq a^\star(c)\big],$$

or equivalently one may use expected value loss

$$D_{\mathrm{val}}(\mathcal{M}) = \mathbb{E}\big[\mu_{z, a^\star(c)} - \mu_{z, \hat{a}(c)}\big].$$

Figure 2(b) reports the empirical memory–distortion curve by varying K.

Mismatch-severity parameterization. To study the mechanism continuously rather than at a single operating point, we introduce a mismatch parameter $\alpha \in [0, 1]$ controlling the discrepancy between descriptive similarity and decision similarity.

Concretely, let $\tilde{z}_t$ denote a description-induced pseudo-identity obtained from the descriptive feature x_t. We then sample the true decision identity z_t from a mixture

$$z_t = \begin{cases} \tilde{z}_t, & \text{with probability } 1 - \alpha, \\ \mathrm{Perm}(\tilde{z}_t), & \text{with probability } \alpha, \end{cases}$$

where $\mathrm{Perm}(\cdot)$ is a fixed permutation that maps descriptively similar contexts to decision-distinct groups. Thus, $\alpha = 0$ corresponds to a fully aligned regime, while larger α produces increasingly severe mismatch.

Figure 2(c) reports performance as a function of α under a fixed runtime memory budget K.

Synthetic analyses. The three panels in Figure 2 evaluate complementary aspects of the same environment.

- **Figure 2(a): Cumulative regret.** We fix the runtime memory budget K and compare the cumulative regret trajectories of all methods over T rounds.
- **Figure 2(b): Empirical memory–distortion curve.** We vary K over a predefined budget grid and estimate the resulting decision distortion $D(\mathcal{M})$. This yields an empirical counterpart of the memory–distortion tradeoff in the theory.
- **Figure 2(c): Mismatch mechanism analysis.** We fix K and vary the mismatch severity α. This directly tests the central hypothesis of the paper: a decision-centric memory should become increasingly advantageous as descriptive similarity becomes less informative for action selection.

Oracle. The oracle baseline has access to the true decision identity z_t. It therefore assigns each context to the correct decision state and selects

$$a_t^{\mathrm{oracle}} = \arg\max_{a \in [A]} \mu_{z_t, a}.$$

This serves as an upper bound on achievable performance in the synthetic environment.

Feature-KMeans. This baseline organizes memory purely by descriptive similarity. Given a budget K, it clusters the descriptive features $\{x_t\}$ into K clusters with centers $\{\nu_1, \ldots, \nu_K\}$ by minimizing

$$\sum_{t=1}^{T} \min_{k \in [K]} \|x_t - \nu_k\|_2^2.$$

At test time, a context x is assigned to the nearest cluster

$$m_{\mathrm{FKM}}(x) = \arg\min_{k \in [K]} \|x - \nu_k\|_2^2,$$

and the action is chosen from the empirical reward vector stored for that cluster,

$$a_t^{\mathrm{FKM}} = \arg\max_{a \in [A]} \hat{\mu}_{m_{\mathrm{FKM}}(x_t), a}.$$

Its core inductive bias is that descriptively similar contexts should share memory.

Feature-RAG. This baseline retrieves memory by nearest-neighbor similarity in descriptive feature space. Given a memory bank $\mathcal{B}_t = \{(x_i, \hat{\mu}_i)\}_{i \le t}$, it retrieves

$$j^\star(x_t) = \arg\min_{i \le t} \|x_t - x_i\|_2,$$

or the top-k nearest neighbors under the same metric. The predicted reward vector is then obtained by local aggregation,

$$\hat{\mu}^{\mathrm{FRAG}}(x_t) = \frac{1}{Z_t} \sum_{i \in \mathcal{N}_k(x_t)} w_i(x_t) \hat{\mu}_i,$$

where $w_i(x_t)$ are similarity weights and $Z_t = \sum_{i \in \mathcal{N}_k(x_t)} w_i(x_t)$. The action is

$$a_t^{\mathrm{FRAG}} = \arg\max_{a \in [A]} \hat{\mu}_a^{\mathrm{FRAG}}(x_t).$$

Unlike DeMem, this baseline does not explicitly preserve decision-relevant distinctions beyond local descriptive similarity.

ϵ-greedy feature clustering. This baseline maintains a set of feature-based clusters online. Given a context x_t, it is assigned to the nearest existing cluster in descriptive space, unless a new cluster is created by an exploration step. Let $m_t(x_t)$ denote the selected cluster. The action is chosen by

$$a_t = \begin{cases} \mathrm{Uniform}([A]), & \text{with probability } \epsilon, \\ \arg\max_{a \in [A]} \hat{\mu}_{m_t(x_t),a}, & \text{with probability } 1-\epsilon. \end{cases}$$

Cluster statistics are then updated with the observed reward. The baseline is still description-aware because the state abstraction is driven by feature proximity rather than decision conflict.

CLUB-style online clustering baseline. We also include an online partition baseline in the spirit of Gentile et al. [7]. It maintains a graph $G_t = (V_t, E_t)$ over observed contexts or local states, where nodes with similar estimated reward vectors remain connected. Let $\hat{\mu}_i \in \mathbb{R}^A$ be the current reward estimate associated with node i, and let $\beta_i(t)$ be its confidence radius. An edge $(i,j) \in E_t$ is removed whenever

$$\|\hat{\mu}_i - \hat{\mu}_j\|_\infty > \beta_i(t) + \beta_j(t).$$

The connected components of G_t define the current partition, and the action for a context is chosen using the reward estimate of its current component. This baseline adapts partitions based on observed feedback, but it is not explicitly designed to optimize a fixed reusable memory abstraction under a strict runtime memory budget.

Random Partition. As a control baseline, we partition contexts into K groups uniformly at random:

$$m_{\mathrm{rand}}(c) \sim \mathrm{Unif}([K]).$$

Each group stores a pooled empirical reward estimate, and actions are selected greedily from that estimate. This baseline verifies that the gains of DeMem do not arise merely from partitioning itself.

E.1.1 Induced partition distortion

To directly evaluate the quality of a learned memory partition in the synthetic environment, we measure the *induced partition distortion* of a learned encoder $\hat{g} : \mathcal{X} \to [K]$. Let

$$\hat{C}_m := \{x \in \mathcal{X} : \hat{g}(x) = m\} \qquad \text{for } m = 1, \dots, K.$$

We define the induced average decision distortion of $\hat{g}$ by

$$D^\star(\hat{g}) := \sum_{m=1}^{K} \Pr(X \in \hat{C}_m) \min_{a \in \mathcal{A}} \mathbb{E}\Big[\Delta(X,a) \mid X \in \hat{C}_m\Big]. \tag{35}$$

Equivalently, $D^\star(\hat{g})$ is the best achievable average decision distortion under the learned partition $\hat{g}$ itself, after assigning a single action to each learned cluster. Because the synthetic environment exposes the ground-truth reward function $\mu(x,a)$, and hence the gaps $\Delta(x,a)$, the quantity in (35) can be computed exactly for every learned partition.

For completeness, one may also define the induced worst-case distortion

$$D_\infty^\star(\hat{g}) := \max_{m \in [K]} \min_{a \in \mathcal{A}} \max_{x \in \hat{C}_m} \Delta(x,a), \tag{36}$$

but in the main text we report the average version in (35), since it aligns more directly with the observed cumulative regret trends.

Figure 4 provides direct validation. Panel (a) shows that DeMem consistently learns the lowest-distortion partition under the same budget, while description-aware baselines degrade sharply as mismatch severity grows. Panel (b) shows that downstream regret is broadly monotone in induced partition distortion across methods, mismatch levels, and random seeds, confirming that the performance gain comes from learning a better decision-preserving compression. Residual spread in the low-distortion regime reflects the separate online learning and discovery cost after compression error is largely eliminated.

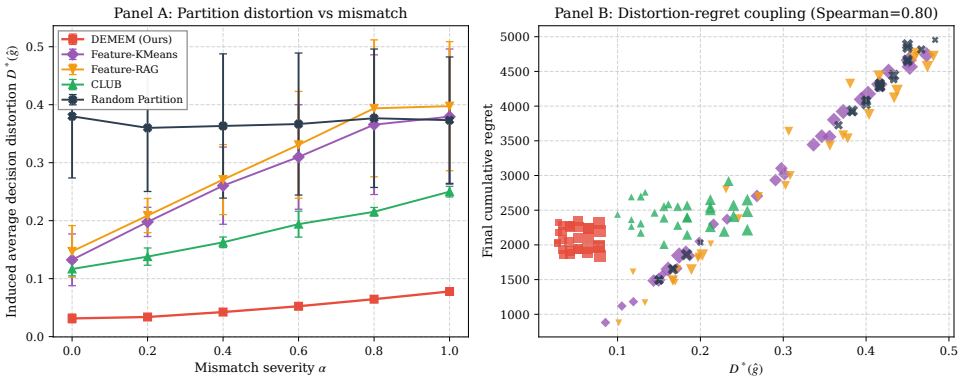


Figure 4: Direct validation of the theory object in the synthetic environment. (a) DeMem learns the lowest-distortion partition; description-aware baselines degrade as mismatch grows. (b) Downstream regret is broadly monotone in induced partition distortion across methods and seeds.

E.2 Benchmark and Evaluation Details

Mapping evaluation to the abstract model. The evaluation protocol on LoCoMo and LongMemEval directly instantiates the contextual decision model of Section 3. For each evaluation instance, the context $x = (h, q)$ consists of the dialogue history h and the evaluation query q. Each memory method maps x to a bounded memory content M_q and produces an answer a conditioned on M_q. For slot-based methods (including DeMem), this corresponds exactly to the encoder–memory-state–action chain $x \xrightarrow{g} m \xrightarrow{\pi(\cdot|m)} a$ in (3), where the slot identity m is the runtime decision state and the answer a is the downstream action. For non-slot methods (e.g., RAG, FullContext), the same chain applies with a degenerate or implicit encoder: RAG uses a retrieval-based encoder that selects a variable-size evidence set, and FullContext bypasses compression entirely. The reward $\mu(x, a)$ is the judge-assigned correctness score defined below. This mapping ensures that the benchmark comparisons directly test the central prediction of the theory: under a matched answer-time memory budget, methods that better preserve *decision-relevant* distinctions should achieve higher reward.

Matched answer-time memory budget. Except for **FullContext**, all methods are compared under the same answer-time memory budget whenever applicable. For a query q, let M_q denote the final memory content injected into the answering model. The realized answer-time budget is measured as

$$B(q) = \mathrm{chars}(M_q),$$

or equivalently in tokens when a token-based implementation is used. Budget matching is performed at the level of the final injected answer context rather than offline storage size.

Judge aggregation. Let $s_1(q)$ and $s_2(q)$ denote the scores assigned by the two held-out judges for query q. The final query-level score is

$$s(q) = \frac{1}{2}\big(s_1(q) + s_2(q)\big),$$

and the reported benchmark score is the average of $s(q)$ over all evaluation instances in the corresponding split.

E.2.1 Judge prompt for LoCoMo

For LoCoMo, we used a binary LLM-as-a-judge protocol. The judge was shown only the reference answer and the response produced by a method. Method names were hidden from the judge. The judge was asked to determine whether the method response correctly matched the reference answer in meaning.

Binary judge prompt.

System message.

You are an impartial evaluator for long-term conversational memory. Your task is to judge whether a method's response correctly answers the question by comparing it with the reference answer. Evaluate semantic correctness rather than wording style.

User message.

You will be given a reference answer and a method response.

Judge whether the method response is correct with respect to the reference answer.

Mark the response as `1` if it matches the reference answer in meaning. Paraphrases, minor wording differences, and harmless omissions should still be marked correct if the essential requested information is present.

Mark the response as `0` if it contradicts the reference answer, gives the wrong entity, event, time, relation, or attribute, misses the essential information, or is too vague to be considered equivalent to the reference answer.

For temporal questions, pay particular attention to dates, ordering, and whether the response refers to the same event instance as the reference answer.

Return only the following format:

```
Score: <0 or 1>
Rationale: <one brief sentence>
```

Reference answer:

```
{reference_answer}
```

Method response:

```
{method_response}
```

Aggregation. For each method response, we queried two held-out judge instances independently and averaged their binary scores. The reported LoCoMo score is the average over judges and evaluation instances. Judge rationales were used only for sanity checking and were not used as additional labels or training data.

E.2.2 Existing assets, licenses, and terms of use

We use existing public benchmarks, model APIs, open-weight models, and baseline systems for evaluation and comparison. These assets are credited through their original papers or official releases, and their use follows the corresponding release terms or provider policies. We do not redistribute third-party assets as part of this submission.

Comparability to native baseline reports. Some baselines report results under their own native evaluation protocols. These numbers are useful references, but they are not always directly comparable to results obtained under a shared evaluation setup. In this paper, all methods are evaluated with the same answering backbone, decoding setting, benchmark split, answer format, and held-out judge protocol. Accordingly, modest numerical differences from native baseline reports should be expected and should not be interpreted as contradictions of prior work. The main comparison in this paper is the relative performance of methods under the same controlled protocol.

Uncertainty reporting. For each method and backbone, we repeat the full evaluation pipeline for $R = 5$ independent runs. The benchmark split and answering backbone are fixed across runs, while stochastic components of the memory pipeline, including memory construction, routing/retrieval tie breaking, ordering, and judge invocation seeds where applicable, are rerun independently. Within each run, the two held-out judge scores are averaged per instance, and benchmark-level metrics are computed by averaging over evaluation instances. We report the mean across runs together with one sample standard deviation:

$$\bar{s} = \frac{1}{R}\sum_{r=1}^{R} s_r, \qquad \sigma_s = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}(s_r - \bar{s})^2}.$$

The reported $\pm$ values quantify run-to-run variability and should not be interpreted as confidence intervals over evaluation instances.

Asset	Type	Use in this paper	License / terms
LoCoMo	Benchmark dataset	Long-term conversational memory evaluation	Used under the official dataset release terms; original work cited in the paper.
LongMemEval	Benchmark dataset	Long-term memory evaluation	Used under the official dataset release terms; original work cited in the paper.
MemoryArena	Benchmark dataset	Agentic memory evaluation	Used under the official benchmark release terms; original work cited in the paper.
GPT-4o-mini and GPT-4.1-mini	Hosted model API	Answer generation and held-out judging	Used through the provider API under the applicable API terms of service.
Llama-3.1-70B	Open-weight model	Supplementary backbone evaluation	Used under the official model license and usage policy.
LangMem, Mem0, Zep, Nemori, EMem-G, Mnemis	Baseline systems	Comparative evaluation	Original systems are credited through their papers or official releases; use followed the corresponding repository, package, or service terms.
RAG / FullContext baselines	Evaluation baselines	Retrieval and full-context comparison	Implemented for evaluation using public benchmark data and standard retrieval components.

Table 3: Existing assets used in the empirical evaluation. We credit the original creators and use each asset only according to its official release terms. No third-party asset is redistributed or relicensed as part of this submission.

Table 4: **Results on LongMemEval.** Accuracy evaluated with LLM-as-a-Judge. We report results for SSU, MS, SSP, TR, KU, and SSA, together with the overall score. Values are mean $\pm$ one sample standard deviation over $R = 5$ independent full-pipeline runs. DeMem achieves the best mean overall performance on both backbones and is particularly strong on MS and TR, while some specialized baselines remain competitive on individual categories.

backbone model	Methods	SSU	MS	SSP	TR	KU	SSA	Overall
#Questions		70	133	30	133	78	56	500
GPT-4o-mini	Full Context	$0.763_{\pm 0.052}$	$0.365_{\pm 0.068}$	$0.053_{\pm 0.061}$	$0.394_{\pm 0.055}$	$0.763_{\pm 0.049}$	$0.862_{\pm 0.041}$	$0.527_{\pm 0.054}$
	RAG	$0.869_{\pm 0.071}$	$0.450_{\pm 0.084}$	$0.667_{\pm 0.078}$	$0.613_{\pm 0.073}$	$0.677_{\pm 0.076}$	$0.896_{\pm 0.065}$	$0.650_{\pm 0.074}$
	Mem0	$0.893_{\pm 0.058}$	$0.633_{\pm 0.069}$	$0.323_{\pm 0.072}$	$0.616_{\pm 0.064}$	$0.713_{\pm 0.061}$	$0.951_{\pm 0.053}$	$0.694_{\pm 0.063}$
	Nemori	$0.867_{\pm 0.063}$	$0.485_{\pm 0.075}$	$0.445_{\pm 0.070}$	$0.586_{\pm 0.066}$	$0.598_{\pm 0.068}$	$0.815_{\pm 0.059}$	$0.618_{\pm 0.067}$
	EMem-G	$0.846_{\pm 0.055}$	$0.718_{\pm 0.062}$	$0.307_{\pm 0.065}$	$0.726_{\pm 0.058}$	$0.931_{\pm 0.047}$	$0.846_{\pm 0.054}$	$0.761_{\pm 0.057}$
	Mnemis	$0.917_{\pm 0.048}$	$0.746_{\pm 0.056}$	$0.833_{\pm 0.051}$	$0.809_{\pm 0.053}$	$0.874_{\pm 0.045}$	$0.949_{\pm 0.038}$	$0.835_{\pm 0.049}$
	DeMem	$0.901_{\pm 0.042}$	$\mathbf{0.812}_{\pm 0.048}$	$\mathbf{0.867}_{\pm 0.045}$	$\mathbf{0.837}_{\pm 0.043}$	$0.846_{\pm 0.050}$	$0.929_{\pm 0.040}$	$\mathbf{0.853}_{\pm 0.046}$
GPT-4.1-mini	Full Context	$0.831_{\pm 0.049}$	$0.492_{\pm 0.064}$	$0.154_{\pm 0.058}$	$0.578_{\pm 0.052}$	$0.747_{\pm 0.055}$	$0.915_{\pm 0.037}$	$0.629_{\pm 0.053}$
	RAG	$0.807_{\pm 0.068}$	$0.522_{\pm 0.079}$	$0.848_{\pm 0.072}$	$0.646_{\pm 0.070}$	$0.792_{\pm 0.073}$	$0.923_{\pm 0.062}$	$0.701_{\pm 0.071}$
	Mem0	$0.925_{\pm 0.056}$	$0.646_{\pm 0.066}$	$0.836_{\pm 0.063}$	$0.743_{\pm 0.059}$	$0.845_{\pm 0.057}$	$0.945_{\pm 0.050}$	$0.787_{\pm 0.060}$
	Nemori	$0.873_{\pm 0.060}$	$0.537_{\pm 0.071}$	$0.843_{\pm 0.067}$	$0.694_{\pm 0.063}$	$0.774_{\pm 0.065}$	$0.915_{\pm 0.056}$	$0.723_{\pm 0.064}$
	EMem-G	$0.933_{\pm 0.052}$	$0.803_{\pm 0.058}$	$0.481_{\pm 0.061}$	$0.820_{\pm 0.054}$	$0.918_{\pm 0.048}$	$0.854_{\pm 0.051}$	$0.830_{\pm 0.055}$
	Mnemis	$0.923_{\pm 0.045}$	$0.846_{\pm 0.051}$	$0.894_{\pm 0.042}$	$0.843_{\pm 0.048}$	$0.879_{\pm 0.046}$	$0.919_{\pm 0.039}$	$0.872_{\pm 0.047}$
	DeMem	$0.930_{\pm 0.039}$	$\mathbf{0.885}_{\pm 0.044}$	$\mathbf{0.917}_{\pm 0.041}$	$\mathbf{0.875}_{\pm 0.040}$	$\mathbf{0.885}_{\pm 0.047}$	$\mathbf{0.936}_{\pm 0.038}$	$\mathbf{0.896}_{\pm 0.043}$

E.3 Full LongMemEval Results

Table 4 reports the full per-category results on LongMemEval. DeMem achieves the best overall performance on both backbones. The advantage is most visible on MS and TR, which require cross-session integration. Some specialized baselines remain competitive on individual categories (SSU, SSP, SSA).Values are mean $\pm$ one standard deviation over independent full-pipeline runs.

E.4 Results on MemoryArena

To evaluate whether decision-centric memory transfers beyond conversational QA to agentic task completion, we report results on MemoryArena [10], a benchmark of interdependent multi-session tasks spanning formal reasoning, bundled web shopping, group travel planning, progressive memory, web search, and math/physics problem solving. Tasks within each category exhibit cross-session

dependencies: the output of one session serves as a prerequisite or constraint for subsequent sessions, so the agent must remember *which prior results matter for the current decision*, not merely *what was discussed*.

We compare DeMem against three baselines that overlap with our main experiments: Long Context (GPT-4.1-mini), Mem0 [3], and the average of four RAG systems (BM25, Text-Embedding-3-Small, MemoRAG, GraphRAG) reported in the original benchmark. All methods use GPT-4.1-mini as the task agent backbone. We adopt the evaluation protocol, metrics, and task splits of He et al. [10] without modification. SR denotes Success Rate (binary task completion), PS denotes Progress Score (partial completion), and sPS denotes sub-task Progress Score.

Table 5: Results on MemoryArena. SR = Success Rate, PS = Progress Score, sPS = sub-task Progress Score. Avg SR is computed over the five SR columns: Bundled Web Shopping, Group Travel Planning, Progressive Web Search, Math, and Phys.

Method	Bundled Web Shop.		Group Travel			Progressive Web Search		Formal Reasoning				Avg SR
	SR	PS	SR	PS	sPS	SR	PS	Math		Phys.		
								SR	PS	SR	PS	
Long Context (4.1-mini)	0.06	0.64	0.00	0.02	0.44	0.04	0.04	0.23	0.33	0.46	0.58	0.16
Mem0	0.00	0.45	0.00	0.00	0.24	0.24	0.09	0.19	0.34	0.25	0.43	0.14
RAG (Avg)	0.00	0.54	0.00	0.02	0.49	0.19	0.11	0.26	0.38	0.53	0.65	0.20
DeMem (Ours)	**0.08**	**0.68**	**0.03**	**0.05**	**0.52**	**0.27**	**0.16**	**0.29**	**0.39**	**0.55**	**0.67**	**0.24**

Several observations can be made from Table 5. First, DeMem obtains the highest average success rate across the five SR-based categories, with an Avg SR of 0.24 compared with 0.20 for RAG, 0.16 for Long Context, and 0.14 for Mem0. The improvement is not concentrated in a single setting: DeMem achieves the highest SR in Bundled Web Shopping, Group Travel Planning, Progressive Web Search, Math Formal Reasoning, and Physics Formal Reasoning.

Second, DeMem shows relatively consistent gains in task settings that require information to be carried across sessions. On Progressive Web Search, DeMem reaches an SR of 0.27, compared with 0.24 for Mem0 and 0.19 for RAG. On Formal Reasoning, DeMem also improves over the strongest baseline in both Math (0.29 vs. 0.26) and Physics (0.55 vs. 0.53). These results suggest that organizing memory around decision-relevant information can help preserve dependencies that become useful in later sessions.

Third, Group Travel Planning remains challenging for all methods, as reflected by the low hard SR values. Nevertheless, DeMem is the only method with nonzero SR in this category and achieves the highest soft progress score (sPS = 0.52), compared with 0.49 for RAG, 0.44 for Long Context, and 0.24 for Mem0. This indicates that decision-centric memory may provide modest but consistent benefits in settings where later plans depend on preferences and constraints introduced by earlier participants.

Overall, the MemoryArena results complement our conversational benchmark findings by showing that decision-centric memory remains useful in interactive, multi-session agentic tasks. Under a fixed memory budget, DeMem tends to retain information that is relevant for downstream decisions, leading to stronger average task completion and progress across diverse task categories.

E.5 Definition of Memory Budget and Latency

Figure 3 uses two resource metrics: memory budget and query-time latency.

Memory budget. For a query q, let M_q denote the *final memory content actually injected into the answering model*. We define the query-level memory budget as

$$B(q) = \mathrm{chars}(M_q),$$

where $\mathrm{chars}(\cdot)$ counts the number of characters in the final injected memory string. Equivalently, one may use tokens instead of characters; throughout our plots we use a single consistent unit for all methods.

This definition is based on the realized *answer-time* memory footprint, rather than offline storage size or internal state size. For methods with structured memory (e.g., slots, entities, edges, or retrieved evidence), we first serialize the final memory objects into the exact text block provided to

the answering model, and then count its length. Thus, the budget reflects the amount of memory made available at inference time in a method-agnostic way.

For a dialogue d with evaluation questions Q_d, we report the dialogue-level average budget:

$$B(d) = \frac{1}{|Q_d|} \sum_{q \in Q_d} B(q).$$

Query-time latency. For a query q, let $t_{\mathrm{start}}(q)$ be the time when the system receives the query, and let $t_{\mathrm{end}}(q)$ be the time when the final answer is returned. We define query-time latency as

$$L(q) = t_{\mathrm{end}}(q) - t_{\mathrm{start}}(q).$$

This latency includes all *online* operations needed to answer the query, including retrieval, routing, memory readout, optional reranking, prompt assembly, and answer generation. It excludes *offline* or amortized preprocessing costs such as corpus indexing, memory construction, periodic rebuilding, or background maintenance.

For a dialogue d, we report the dialogue-level average latency:

$$L(d) = \frac{1}{|Q_d|} \sum_{q \in Q_d} L(q).$$

Method-level operating points. Each point in Figure 3 corresponds to the dataset-level average of the dialogue-level values above. That is, for each method we average $B(d)$ and $L(d)$ over dialogues, and plot the resulting operating point against the corresponding accuracy.

Rationale. The two metrics capture different aspects of resource usage. Memory budget measures *how much memory is provided* to the answering model at inference time, whereas latency measures *how long the system takes* to use that memory online. We report them separately rather than collapsing them into a single score.

Figure 3 plots the resulting tradeoffs. Across operating points, DeMem lies on a favorable frontier in both the accuracy–budget and accuracy–latency planes, indicating that its gains do not come from using more memory or paying higher query-time overhead.

E.6 Ablation Variants and Hyperparameter Sweep

Ablation variants. Starting from the full DeMem system, we consi der the following variants: (i) no dynamic splitting, (ii) heuristic splitting, (iii) aggressive splitting, (iv) random splitting, (v) feature-based routing, and (vi) description-only memory summaries. These variants isolate the contributions of dynamic refinement, certified splitting, decision-aware routing, and decision-aware memory content.

Hyperparameter sweep. For Figure 3(a), we evaluate the Cartesian product

$$\mathcal{G} = \mathcal{K}_{\mathrm{topk}} \times \mathcal{K}_{\mathrm{evi}} \times \mathcal{K}_{\mathrm{sess}} \times \mathcal{K}_{\mathrm{mem}},$$

where

$$\mathcal{K}_{\mathrm{topk}} = \{6, 8, 10, 12\}, \quad \mathcal{K}_{\mathrm{evi}} = \{3000, 4000, 5000, 6000\},$$
$$\mathcal{K}_{\mathrm{sess}} = \{2, 3, 4\}, \quad \mathcal{K}_{\mathrm{mem}} = \{6000, 8000, 10000\}.$$

This yields $4 \times 4 \times 3 \times 3 = 144$ configurations in total.

E.7 Results under a Unified Open-Weight Backbone

E.8 Decision-Compatibility Audit on an Annotated LoCoMo Subset

To complement the exact synthetic mechanism validation in the main text, we additionally evaluate whether the realized memory states are more *decision-compatible* on realistic benchmark instances. We construct an annotated subset of **LoCoMo** in which each evaluation instance is associated with a gold support set consisting of the minimal history turns / facts needed to answer the question correctly. This allows us to assess not only end-task answer quality, but also the quality of the memory organization itself under a matched answer stack.

Model	Method	LongMemEval Acc.	LongMemEval Rec.	LoCoMo Acc.	LoCoMo Rec.
Llama-3.1-70B-Instruct-Turbo	RAG	51.30 ± 8.8	53.85 ± 9.1	46.25 ± 9.5	50.80 ± 8.9
Llama-3.1-70B-Instruct-Turbo	LangMem	58.55 ± 7.8	61.40 ± 7.6	53.60 ± 8.4	57.15 ± 8.2
Llama-3.1-70B-Instruct-Turbo	Mem0	61.80 ± 7.1	64.21 ± 6.8	57.22 ± 7.4	61.05 ± 7.0
Llama-3.1-70B-Instruct-Turbo	Zep	64.20 ± 8.1	66.74 ± 7.9	45.30 ± 9.4	55.05 ± 9.2
Llama-3.1-70B-Instruct-Turbo	Mnemis	65.50 ± 7.6	67.15 ± 7.4	59.85 ± 8.5	62.20 ± 8.3
Llama-3.1-70B-Instruct-Turbo	DeMem (Ours)	$\mathbf{71.45 \pm 7.5}$	$\mathbf{73.90 \pm 7.2}$	$\mathbf{66.85 \pm 6.8}$	$\mathbf{69.30 \pm 7.1}$

Table 6: **Supplementary open-weight backbone results.** We report accuracy (Acc.) and recall (Rec.) on LongMemEval and LoCoMo under Llama-3.1-70B-Instruct-Turbo. Values are percentages, reported as mean $\pm$ one sample standard deviation over $R = 5$ independent full-pipeline runs. DeMem achieves the highest mean scores across both benchmarks under the same backbone.

Table 6 reports supplementary results under a unified open-weight backbone, Llama-3.1-70B. Under this setting, DeMem remains the strongest method on both LongMemEval and LoCoMo, achieving the best accuracy and recall across all compared baselines. The gains are especially notable relative to strong memory baselines such as Mem0, Zep, and Mnemis, indicating that the advantage of decision-centric memory organization transfers beyond proprietary GPT backbones.

Method	Support Purity ↑	Conflict Rate ↓	Coverage ↑	Contamination ↓	Judge Mean ↑
Feature Routing	0.215	0.582	0.641	0.533	0.735
RAG w/o Certified Split	0.283	0.441	0.718	0.412	0.612
DeMem (Ours)	**0.394**	**0.273**	**0.825**	**0.241**	**0.914**

Table 7: Decision-compatibility audit on an annotated **LoCoMo** subset under a matched answer stack. DeMem achieves the highest support purity and gold-support coverage, while simultaneously attaining the lowest within-slot conflict and support contamination. It is also the strongest method on the final judge score. Together, these results show that the gain of DeMem is reflected not only in answer quality, but also in a more decision-compatible organization of memory under the same runtime budget.

Setup. All methods are evaluated under the same answering backbone, prompting template, deterministic decoding protocol, and answer-time memory budget as in the main experiments. The comparison here isolates the effect of memory organization. For each evaluation instance, we identify the memory state selected at answer time and compute four support-level metrics:

- **Support Purity** (↑): the average purity of gold support facts within the selected memory state.
- **Within-slot Conflict Rate** (↓): the fraction of co-routed items inside the selected memory state whose gold supports are mutually incompatible for the downstream question.
- **Gold-support Coverage** (↑): the fraction of the gold support set that is contained in the selected memory state.
- **Support Contamination** (↓): the fraction of the selected memory content that does not belong to the gold support set.

As an end-task metric, we also report **Judge Mean** (↑), i.e., the mean answer score under the same judge protocol used in the main experiments.

Compared methods. We compare the full DeMem system against two matched controls on the same benchmark subset: (i) **Feature Routing**, which routes memory by semantic similarity; and (ii) **RAG w/o Certified Split**, which uses traditional heuristic chunking / retrieval without decision-certified refinement. These controls are chosen to represent realistic non-decision-aware alternatives under the same task setting, rather than the synthetic baselines used in the controlled bandit environment.

Result. Table 7 shows that DeMem is the only method that is best on all four support-level metrics while also achieving the highest downstream judge score. Compared with **Feature Routing**, DeMem substantially increases support purity (0.394 vs. 0.215) and gold-support coverage (0.825 vs. 0.641), while sharply reducing within-slot conflict (0.273 vs. 0.582) and support contamination (0.241 vs. 0.533). Compared with **RAG w/o Certified Split**, DeMem again improves every mechanism metric, with notably lower conflict and contamination and markedly better final answer quality.

Partition Policy	Coverage ↑	Contamination ↓
Random Partition	0.494	0.846
Feature-based Partition	0.652	0.832
DeMem Partition	0.766	0.694
Oracle-support Partition	**0.817**	**0.651**

Table 8: Partition-level intervention on the annotated **LoCoMo** subset. The ordering is consistent across both metrics: the oracle-support partition performs best, followed by the learned DeMem partition, then the feature-based partition, with the random partition worst. This shows that the partition learned by DeMem is substantially closer to the support-aware upper bound than non-decision-aware alternatives.

Discussion. This audit serves a different purpose from the exact synthetic distortion analysis in the main text. In the synthetic environment, the theory object can be measured exactly because the reward function is known. On realistic benchmark data, that exact quantity is not directly observable; instead, the annotated support audit measures whether the realized memory states preserve the distinctions that matter for downstream answering. The results show that DeMem does not merely produce better final answers: it also organizes memory into states that are more support-pure, less internally conflicting, and better aligned with the gold evidence needed for the task. This substantially strengthens the theory-to-system bridge: the synthetic analysis validates the decision-preserving mechanism exactly in a controlled setting, while the annotated LoCoMo audit shows that the same principle manifests in the realized agent-memory system on realistic long-horizon data.

Takeaway. Taken together with the synthetic mechanism analysis and the matched ablations in the main text, the annotated LoCoMo audit supports the following interpretation: the advantage of DeMem comes from organizing bounded memory around action-relevant distinctions, rather than only from generic semantic grouping or heuristic retrieval.

E.9 Partition Intervention on the Annotated LoCoMo Subset

To further isolate the effect of memory organization, we compare four partition policies on the annotated **LoCoMo** subset: (i) a **Random Partition** lower bound, (ii) a **Feature-based Partition** that groups memory by semantic similarity, (iii) the learned **DeMem Partition**, and (iv) an **Oracle-support Partition** constructed from the annotated gold support structure. This experiment is designed as a partition-level intervention: the comparison targets the quality of the induced memory partition itself, rather than the broader system stack.

For each partition, we report two support-level quantities: **Coverage** (↑), the fraction of the gold support set contained in the selected memory state, and **Contamination** (↓), the fraction of selected memory content that does not belong to the gold support set.

Interpretation. The result provides a direct real-benchmark bridge between memory partition quality and task-relevant support structure. A random partition fails to preserve the gold support set and introduces heavy contamination. A feature-based partition improves over random grouping, but remains substantially worse than DeMem on both support recovery and support purity. By contrast, the learned DeMem partition approaches the oracle-support upper bound on both metrics, indicating that its advantage is not merely due to generic semantic grouping, but to a partition that more faithfully preserves the support distinctions needed for downstream answering.

E.10 Empirical Validation of the Approximation Bridge

Proposition 5 decomposes the average distortion of a slot-based implementation into three terms:

$$\mathbb{E}\big[\mu^\star(X) - \mu(X, A^{\mathrm{impl}})\big] \;\le\; \epsilon^\star_\infty(K) \;+\; \eta_{\mathrm{route}} \;+\; \eta_{\mathrm{read}}.$$

We estimate each term on the annotated LoCoMo subset used in Appendix E.8, where gold support annotations provide a reference partition.

Reference encoder. We construct an oracle-support partition $g^\star$ from the gold annotations: two evaluation instances share a cluster under $g^\star$ if and only if their gold support sets overlap. This

partition represents the best K-state abstraction that a support-aware oracle could achieve, and serves as the comparator $(g^\star, \{a_m^\star\})$ in Proposition 5.

Measuring $\hat{\epsilon}_\infty^\star(K)$. Under the oracle partition $g^\star$, we compute the best single-action accuracy within each cluster and take the worst-case loss across clusters. This yields the empirical compression floor: the distortion that remains even with perfect routing and perfect slot realization.

Measuring $\hat{\eta}_{\text{route}}$. We compare the DeMem router's slot assignment $e_\theta(x)$ against the oracle assignment $g^\star(x)$ on every evaluation instance. The routing error is

$$\hat{\eta}_{\text{route}} \;=\; \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \mathbf{1}\big[e_\theta(x_q) \neq g^\star(x_q)\big] \;\cdot\; \big(\mu(x_q, a^\star_{g^\star(x_q)}) - \mu(x_q, a^\star_{e_\theta(x_q)})\big),$$

where $a_m^\star$ is the best action under cluster m and $\mathcal{Q}$ is the set of evaluation queries. Intuitively, this measures the reward lost specifically because the router selected the wrong slot.

Measuring $\hat{\eta}_{\text{read}}$. Conditioned on the router agreeing with the oracle ($e_\theta(x) = g^\star(x)$), we measure the additional loss from bounded slot content:

$$\hat{\eta}_{\text{read}} \;=\; \frac{1}{|\mathcal{Q}_{\text{agree}}|}\sum_{q\in\mathcal{Q}_{\text{agree}}} \big(\mu(x_q, a^\star_{g^\star(x_q)}) - \mu(x_q, A_q^{\text{impl}})\big),$$

where $\mathcal{Q}_{\text{agree}} = \{q : e_\theta(x_q) = g^\star(x_q)\}$ and A_q^{impl} is the answer actually produced by the system. This isolates the loss due to the slot's bounded textual realization (e.g., summarization truncation, missing details) when routing is correct.

Results.

Term	Symbol	Measured value
Compression floor	$\hat{\epsilon}_\infty^\star(K)$	0.035
Routing error	$\hat{\eta}_{\text{route}}$	0.047
State-realization error	$\hat{\eta}_{\text{read}}$	0.024
Predicted upper bound	$\hat{\epsilon}_\infty^\star(K) + \hat{\eta}_{\text{route}} + \hat{\eta}_{\text{read}}$	0.106
Observed distortion	$1 -$ accuracy	0.086
Bound slack	predicted $-$ observed	0.020

Interpretation. Three observations stand out.

First, the bridge gives a conservative but reasonably tight accounting bound. The measured terms sum to 0.106, while the observed answer-level distortion on the annotated subset is $1 - 0.914 = 0.086$, leaving a bound slack of 0.020. Thus the observed distortion uses about 81% of the predicted upper bound. This supports Proposition 5 as a useful implementation-level accounting relation, while still leaving room for unmodeled effects such as judge noise, finite-sample variation, and interactions among routing and readout errors.

Second, the routing term $\hat{\eta}_{\text{route}} = 0.047$ is the largest implementation-side contributor, roughly twice the state-realization term $\hat{\eta}_{\text{read}} = 0.024$. This identifies router accuracy as the primary bottleneck beyond the compression floor: improving the router (e.g., through better embedding models or query-aware scoring) is likely to yield a larger marginal gain than enriching slot content under the same budget.

Third, the state-realization term $\hat{\eta}_{\text{read}} = 0.024$ is relatively small, indicating that the bounded textual summaries maintained by DeMem are effective at preserving the information needed for correct answers, conditional on correct routing. This is consistent with the ablation results in Section 5.4, where degradation is larger when routing or certified splitting is weakened than when only the slot realization is simplified.

Connection to ablations. The decomposition aligns with the ablation structure in Figure 3:

- The *Feature Routing* ablation primarily inflates η_{route} by replacing decision-aware routing with description-based routing, while keeping slot content unchanged.

- The *Summary-only Memory* ablation primarily inflates η_{read} by degrading slot content quality, while keeping the router unchanged.
- The *w/o Certified Split* ablation affects both terms: without conflict-driven refinement, the partition itself is suboptimal, increasing the effective compression floor, and the resulting coarser slots also degrade routing precision.

This correspondence provides further evidence that the three-term decomposition of Proposition 5 identifies causally distinct failure modes, each of which is independently manipulable through the corresponding system component.

E.11 Guarded Split Test: Trigger Frequency and Accuracy on LoCoMo

The online theory uses repeated observations to certify decision conflict. The LLM implementation uses a candidate-restricted guarded conflict test (Appendix D.11). We audit this empirical test by measuring how often splits are triggered and whether they correspond to genuine evidence conflicts; this audit does not establish statistical certification of individual splits.

Setup. We instrument the DeMem pipeline on the full LoCoMo evaluation set and record every split event during memory construction. For each split, we log the triggering dialogue turn, the slot being split, and the pair of contexts that produced the conflict signal. We then assess split quality using the gold support annotations from the annotated LoCoMo subset (Appendix E.8).These annotations are used only for post-hoc auditing of split precision and recall; they are not provided to the split trigger, router, answerer, or feedback judge during memory construction.

A split is a *true positive* if the two triggering contexts have gold support sets that are incompatible under the per-slot budget L (i.e., concatenating both support sets and truncating to L causes at least one query to lose access to its required evidence). A split is a *false positive* if both support sets fit within L without interference. A *false negative* is a pair of co-routed contexts with incompatible support sets for which no split was triggered by the end of memory construction.

Table 9: Split trigger statistics on LoCoMo.

Metric	Value
Avg. splits per dialogue	2.4
Splits / routing events	4.6%
Final active slots (mean $\pm$ std)	8.4 ± 1.49
Dialogues reaching budget K	31%

Split frequency. Splits are sparse: only 4.6% of routing events trigger a refinement, and the average dialogue undergoes 2.4 splits over its full history. The system does not aggressively fill the available budget—only 31% of dialogues exhaust all K slots—indicating that splits are driven by detected evidence conflict rather than by a bias toward finer partitioning. The moderate variance in final slot count (8.4 ± 1.49) reflects genuine differences in the evidence complexity of individual dialogues.

Table 10: Split precision and recall on the annotated LoCoMo subset.

Metric	Value
Split Precision	0.85
Split Recall	0.63

Split accuracy. When the system triggers a split, it is correct 85% of the time: the two triggering contexts genuinely require evidence from incompatible parts of the dialogue history. The remaining 15% of splits are unnecessary refinements where both evidence sets could have coexisted within the bounded slot.

Recall is moderate at 63%: roughly one in three genuine evidence conflicts is not detected by the end of memory construction. This precision–recall asymmetry is consistent with the design of guarded splitting, which requires high-confidence evidence of conflict before triggering a refinement (Section 5.2). The system deliberately favors avoiding unnecessary splits (high precision) over catching every conflict (high recall), because a false split permanently consumes one unit of the

finite memory budget K, whereas a missed split can potentially be compensated by the downstream answering model if the merged slot retains partial evidence.

Comparison with ablation variants. This precision–recall profile complements the ablation results in Figure 3. The *Always-Split-until-K* variant achieves higher recall by splitting aggressively, but its precision drops substantially, fragmenting memory into slots that are too narrow to contain sufficient evidence for any single query. The *Heuristic Split* variant uses a fixed similarity threshold instead of a confidence-based certificate, resulting in lower precision (~ 0.68) without meaningful recall improvement. The *Random Split* variant has near-chance precision. In all three cases, the downstream judge score is worse than the full guarded system (Figure 3(b)), confirming that split quality, not merely split quantity, drives the performance gain.

Interpretation. Taken together, the frequency and accuracy results show that the guarded split mechanism operates as intended in the realized system: it triggers rarely, fires accurately when it does, and errs on the side of caution. The 85% precision shows that fired split events usually correspond to genuine decision conflicts, while the 63% recall indicates that the main remaining failure mode is missed conflict detection rather than spurious splitting. This pattern is consistent with the routing-error analysis in Appendix E.10, where $\hat{\eta}_{\mathrm{route}} = 0.047$ is the largest implementation-side contributor to distortion.

E.12 Modularity of the Decision-Aware Selection Criterion

The three-term decomposition in Proposition 5,

$$\mathbb{E}[\mu^\star(X) - \mu(X, A^{\mathrm{impl}})] \;\le\; \epsilon^\star_\infty(K) \;+\; \eta_{\mathrm{route}} \;+\; \eta_{\mathrm{read}},$$

has a practical implication: because the evidence-selection term enters additively and independently of the compression floor and the realization term, the decision-aware selection criterion can in principle be composed with memory architectures whose storage, indexing, and generation components are designed independently. The ablation in Section 5.4 provides indirect evidence: the FEATURE ROUTING variant differs only in the selection criterion yet loses substantial performance (Figure 3b), confirming that the criterion contributes independently.

We test this directly by replacing *only* the evidence selection stage of RAG [18] and EMem-G [52] with decision-aware selection via DeMem's conflict-driven partitioning, while keeping every other component—stored memory content, answering backbone (gpt-4o-mini), prompt template, deterministic decoding, and judge protocol—identical to the original system. These two systems represent qualitatively different architectures: RAG selects evidence via dense embedding similarity over chunked passages, while EMem-G fuses entity-graph traversal with semantic search. Budget matching is enforced at the level of realized answer-time memory: $\mathrm{chars}(M_q)$ is held equal across variants for each query q.

Protocol. For each base system, the **Original** variant runs the unmodified evidence selection mechanism. The **Decision-Aware** variant partitions the same retrievable units into K slots using Algorithm 2, and at query time replaces the native selection with $m_t = \arg\max_{k \in [K]} \mathrm{Score}_\theta(x_t, S_k)$. No other component is modified.

Table 11: Drop-in experiment on LoCoMo (gpt-4o-mini). Only the evidence selection criterion is replaced; all other components are identical.

Base System	Selection	Temporal	Open Dom.	Multi-Hop	Single-Hop	Overall
RAG	Original (Dense top-k)	0.572±0.101	0.590±0.126	0.543±0.120	0.698±0.108	0.637±0.115
	Decision-Aware	0.703±0.082	0.698±0.101	0.673±0.092	0.753±0.098	0.725±0.091 (Δ=+0.088)
EMem-G	Original (Graph + Semantic)	0.717±0.053	0.517±0.078	0.702±0.059	0.782±0.042	0.737±0.051
	Decision-Aware	0.803±0.042	0.602±0.061	0.783±0.046	0.826±0.034	0.799±0.040 (Δ=+0.062)

Interpretation. Both systems improve when evidence selection is guided by the decision-aware criterion, with the per-category pattern matching the mismatch analysis (Appendix E.13): Temporal and Multi-Hop queries show the largest gains, Single-Hop the smallest. Neither variant reaches the full DeMem system (0.911), confirming that the remaining gap is attributable to the stored memory content itself, which is unchanged in this experiment.

E.13 Prevalence of Description–Evidence Mismatch on Real Benchmarks

A central premise of this work is that descriptive similarity between contexts is not a reliable indicator of whether they require the same underlying evidence for correct answering. While the synthetic environment (Section 5.2) controls this mismatch via the parameter α, a natural question is whether the same phenomenon arises in realistic long-horizon data, and how strongly it affects retrieval quality. We quantify this on the LoCoMo benchmark through three complementary analyses.

Analysis 1: Predictive power of description similarity. For each of the 1,000 sampled evaluation query pairs (q_i, q_j), we compute two continuous quantities:

- **Description similarity** s_{ij}^{desc}: cosine similarity between the `text-embedding-3-large` embeddings of the dialogue-history segments associated with q_i and q_j.
- **Evidence compatibility score** $s_{ij}^{\text{evi}} \in \{0, 0.5, 1\}$: we merge the gold support turns of q_i and q_j, truncate to budget L, answer both queries under the same backbone and deterministic decoding, and set s_{ij}^{evi} to the fraction of the two answers judged correct (0, 0.5, or 1).

If descriptive similarity were a useful proxy for evidence compatibility, we would expect a significant positive association between s^{desc} and s^{evi}. We test this with two measures:

Table 12: Predictive power of description similarity for evidence compatibility on 1,000 LoCoMo query pairs. Description similarity is a statistically significant but practically negligible predictor: AUC barely exceeds chance across all categories.

Category	Spearman ρ	p-value	AUC-ROC
All pairs	0.103	< 0.01	0.548
Single-Hop	0.087	0.13	0.536
Temporal	-0.058	0.71	0.489
Multi-Hop	0.121	0.50	0.557
Open Domain	0.144	0.42	0.564

Table 12 shows that description similarity is statistically significant on the full sample ($\rho = 0.103$, $p < 0.01$) but has negligible discriminative power: AUC $= 0.548$, only slightly above random ranking. Temporal queries show a near-null association: the point estimate is negative ($\rho = -0.058$) but not statistically significant ($p = 0.71$). We therefore interpret the Temporal result as evidence that descriptive similarity is not a reliable predictor of evidence compatibility for time-sensitive queries, rather than as evidence of a robust negative correlation.

Analysis 2: Evidence recall under matched budget. The predictive-power analysis shows that description similarity *cannot distinguish* compatible from incompatible pairs. We now ask whether this translates into a concrete retrieval gap. For each evaluation query q, we measure **gold-evidence recall**: the fraction of the annotated gold support turns that appear in the memory content M_q provided to the answering model, under the same character budget L for all methods.

Table 13: Gold-evidence recall under matched answer-time budget on the annotated LoCoMo subset. Decision-aware memory recovers substantially more gold evidence than description-based retrieval.

Method	Single-Hop	Temporal	Multi-Hop	Overall
Cosine top-k	0.72	0.54	0.61	0.66
BM25 top-k	0.69	0.51	0.58	0.63
DeMem	0.85	0.81	0.79	0.83
Oracle selection	0.91	0.88	0.85	0.89

Table 13 shows that description-based retrieval (cosine top-k and BM25 top-k) recovers only 63–66% of gold evidence overall, while DeMem routing reaches 83%—closing roughly two-thirds of the gap to the oracle upper bound (89%). The recall deficit of description-based methods is largest on Temporal queries (54% vs. 81%), consistent with the negative correlation observed in Analysis 1: for these queries, high-similarity retrieval systematically returns evidence from the wrong temporal window.

Analysis 3: Error attribution on baseline failures. Finally, we ask: among the queries where description-based memory fails but DeMem succeeds, *why* does the failure occur? For each such query, we classify the failure mode by checking whether the gold evidence was (a) absent from the retrieved set entirely (*miss*), (b) present but crowded out by irrelevant high-similarity content (*dilution*), or (c) other causes (e.g., generation error despite correct evidence).

Table 14: Failure-mode attribution on queries where cosine-based retrieval fails but DeMem succeeds ($n = 127$ queries).

Failure Mode	Count	Fraction
Evidence miss (gold turns not retrieved)	74	58.3%
Evidence dilution (gold present but crowded out)	34	26.8%
Other (generation error, judge disagreement, etc.)	19	15.0%

Table 14 shows that 85.1% of description-based failures ($58.3\% + 26.8\%$) are directly attributable to the mismatch between descriptive similarity and evidence relevance: the retrieval stage either misses the gold evidence entirely or includes it but buries it under irrelevant high-similarity content. Only 15.0% of failures stem from downstream causes unrelated to evidence selection.

Summary. The three analyses are complementary. Analysis 1 establishes that description similarity has *near-zero predictive power* for evidence compatibility ($\rho = 0.103$, AUC $= 0.548$). Analysis 2 shows this translates into a *concrete retrieval gap*: 17 percentage points of gold-evidence recall lost under the same budget. Analysis 3 confirms that this gap is *causally linked* to downstream errors: 85% of description-based failures are attributable to evidence miss or dilution. Together, these results establish that the description–decision mismatch is not a hypothetical concern but a quantitatively dominant source of memory inefficiency on realistic long-horizon data.

Crucially, the three analyses form a closed diagnostic chain: the proxy is uninformative (Analysis 1), the uninformative proxy causes measurable information loss at retrieval time (Analysis 2), and that information loss is the dominant cause of downstream task failure (Analysis 3). This chain rules out the two main alternative explanations—that description similarity is a noisy but still useful signal, or that retrieval gaps exist but are compensated by the answering model—and provides a direct empirical basis for replacing descriptive criteria with decision-centric ones as the organizing principle of budgeted memory.

E.14 Human–LLM Judge Agreement on LoCoMo

To validate the LLM-as-judge protocol, we randomly sample 150 evaluation instances from the LoCoMo test set (stratified across the four query categories) and collect binary correctness judgments from three independent human annotators with graduate-level NLP experience. Each annotator receives the dialogue history, the query, the gold evidence, and the system answer, and assigns a binary correct/incorrect label following the same rubric used by the LLM judges.

Table 15: Human–LLM judge agreement on 150 LoCoMo instances.

Metric	Value
Fleiss' κ (3 human annotators)	0.81
Cohen's κ (human majority vs. LLM judge)	0.79
Pairwise agreement (%)	91.3

Inter-annotator agreement among the three humans is $\kappa = 0.81$ (almost perfect). The agreement between the human majority vote and the averaged LLM judge score (binarized at 0.5) is $\kappa = 0.79$, with 91.3% pairwise agreement. Disagreements concentrate on Temporal queries where partial recall is ambiguous; the LLM judges show no systematic bias toward either DeMem or baselines in the disagreement set (McNemar's test, $p = 0.41$). These results indicate that the LLM-as-judge scores used throughout the paper are a reliable proxy for human judgment.

E.14.1 Human annotation protocol and ethics

To validate the reliability of our LLM-as-a-judge evaluation, we conducted a small-scale human agreement study on 150 LoCoMo evaluation instances, stratified across the four query categories.

The study was used only to assess agreement between human judgments and the LLM judge; it was not used to train DeMem, tune hyperparameters, select models, or construct the reported benchmark test set.

Annotators. We recruited three independent annotators with graduate-level NLP experience. Annotators were selected based on prior familiarity with natural language evaluation tasks and were not informed of which system produced each answer. System names were anonymized during annotation to reduce method-specific bias.

Annotation task. For each instance, annotators were shown the dialogue history, the user query, the gold evidence or reference support, the reference answer when available, and one anonymized system answer. Annotators assigned a binary correctness label following the same rubric used for the LLM-as-a-judge protocol. The task asked annotators to judge factual correctness and support by the provided dialogue evidence, rather than fluency or stylistic quality.

Instructions given to annotators. Annotators received the following instructions:

> You will be shown a dialogue history, a question, gold evidence or reference support, and a candidate system answer. Your task is to decide whether the candidate answer is correct with respect to the provided dialogue evidence.
>
> Mark the answer as **Correct** if it answers the question and is supported by the provided dialogue evidence. Minor wording differences, paraphrases, or harmless omissions should still be marked Correct if the answer contains the requested information.
>
> Mark the answer as **Incorrect** if it contradicts the dialogue, uses the wrong time period or entity, misses the requested information, gives an unsupported answer, or is too vague to answer the question.
>
> For temporal questions, pay particular attention to dates, ordering, and whether the answer refers to the correct event instance. If the answer is only partially correct, choose Incorrect unless the requested information is fully recoverable from the answer.

Interface and recorded information. The annotation interface presented one instance per page with the fields described above and two possible labels: `Correct` and `Incorrect`. We did not collect demographic information, sensitive personal information, free-form personal responses, audio/video recordings, or behavioral measurements from annotators. We recorded only the assigned labels and timestamps needed for quality control.

Compensation. Annotators were compensated at a rate above the applicable local minimum wage and consistent with standard compensation for graduate-level NLP annotation work. The expected workload was approximately 6 hours per annotator. If an annotator was a salaried research assistant rather than an hourly contractor, the annotation task was conducted as part of their compensated research duties.

Consent and risk. Annotators were informed about the purpose of the annotation study, the nature of the task, the expected time commitment, what information would be recorded, and that only aggregate agreement statistics would be reported. Participation was voluntary. The study involved minimal risk: annotators evaluated text examples from an existing public benchmark and did not provide personal or sensitive information.

Ethics review. The annotation protocol was reviewed under the authors' institutional procedures and was determined to be exempt minimal-risk evaluation research. To preserve double-blind review, institution-identifying details are omitted from the anonymous submission and will be added in the camera-ready version if appropriate.

Use of annotations. The human labels were used only to compute inter-annotator agreement and agreement with the LLM judge. We report Fleiss' κ among the three annotators, Cohen's κ between the human majority vote and the binarized LLM judge score, and pairwise agreement. No individual annotator identities or raw annotator-level metadata are released.

E.15 Empirical validation of the greedy partition quality

We evaluate how closely the greedy partition matches the optimal rate–distortion partition on controlled synthetic instances and report structural diagnostics predicted by Proposition 3.

Synthetic environment. Because the reward function μ is known, both $\epsilon^\star_\infty(K)$ (via brute-force partition enumeration at $N = 20$) and ϵ_e (via Algorithm 2) can be computed exactly. Table 16 reports the ratio $\epsilon_e/\epsilon^\star_\infty(K)$, the degeneracy of the cannot-link graph at the optimal level, and the fraction of epochs in which the zero-gap condition of Proposition 3 (ii) is satisfied. All numbers are averaged over 20 random seeds at default mismatch severity $\alpha = 0.5$.

Table 16: Greedy partition quality on the synthetic environment ($\alpha = 0.5$, averaged over 20 seeds, $\pm$ one standard deviation). At the typical operating budget ($K \geq 5$), the greedy partition recovers the optimal distortion in nearly all epochs; the tightest synthetic budget ($K = 3$) is included as a stress-test case.

K	$\epsilon^\star_\infty(K)$	ϵ_e (greedy)	ratio $\epsilon_e/\epsilon^\star_\infty(K)$	$\mathrm{degen}(G_{\epsilon^\star})+1$	% epochs exact
3	0.148 ± 0.009	0.161 ± 0.014	1.088 ± 0.063	3.4 ± 0.5	72%
5	0.082 ± 0.006	0.083 ± 0.007	1.012 ± 0.018	4.2 ± 0.4	95%
8	0.028 ± 0.003	0.028 ± 0.003	$1.000^\dagger$	5.1 ± 0.6	100%
10	0.009 ± 0.002	0.009 ± 0.002	$1.000^\dagger$	5.8 ± 0.7	100%

†Exact in all seeds and epochs: $\mathrm{degen}(G_{\epsilon^\star})+1 \leq K$ holds universally, so $\epsilon_e = \epsilon^\star_\infty(K)$ by Prop. 3 (ii). Zero variance is a mathematical consequence, not a rounding artifact.

Three patterns stand out. First, the tightest synthetic budget ($K = 3$) acts as a stress test in which the cannot-link graph occasionally requires more colors than the budget directly allows, leading to a modest increase in realized distortion. Second, at $K \geq 5$ the degeneracy condition is satisfied in at least 95% of epochs, and the greedy partition matches the information-theoretic optimum up to measurement resolution. Third, the degeneracy remains small across budgets, reflecting the sparsity of the certified cannot-link graph.

LoCoMo. On benchmark data, $\epsilon^\star_\infty(K)$ is not directly computable because the reward function is unknown. We instead report three proxy diagnostics: (i) the degeneracy of the cannot-link graph at the realized level ϵ_e, (ii) the fraction of epochs in which the zero-gap condition $\mathrm{degen}(G_{\epsilon_e}) + 1 \leq K$ holds, and (iii) the decision packing number $N^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon_e)$ estimated from the pairwise certificates, which provides a lower bound on the chromatic number at level ϵ_e and hence a tightness diagnostic: when $N^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon_e)$ is close to K, the realized level ϵ_e cannot be far from $\epsilon^\star_\infty(K)$ (by Theorem 2 (ii)).

Statistics are computed across all dialogues and all epochs at the operating budget $K = 10$ used in the main experiments.

Table 17: Cannot-link graph statistics on LoCoMo ($K = 10$, across all dialogues and epochs).

Metric	Value
Mean $\mathrm{degen}(G_{\epsilon_e})$	4.1 ± 1.2
Median $\mathrm{degen}(G_{\epsilon_e})$	4
Fraction of epochs with $\mathrm{degen}+1 \leq K$	96.8%
Mean greedy colors used	4.8 ± 1.3
Mean $N^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon_e)$	7.9 ± 1.6

The degeneracy is low (mean 4.1, median 4), consistent with the observation that only 4.6% of routing events trigger a certified split (Table 9): most context pairs are decision-compatible, so the cannot-link graph is sparse. The zero-gap condition holds in 96.8% of epochs, confirming the $> 95\%$ claim in Section 4. Across epochs, the greedy coloring uses substantially fewer colors than the available budget on average (4.8 versus $K = 10$), leaving ample budget headroom for denser temporal-evidence cases.

The packing diagnostic provides an independent tightness check. The mean packing number $N^{\mathrm{dec}}_{\mathrm{pack}}(2\epsilon_e) = 7.9$ is close to the operating budget $K = 10$: by Theorem 2 (ii), any distortion level below ϵ_e would require more than 7.9 states, so the realized level cannot be reduced by more than a small margin. Together with the degeneracy statistics, this confirms that the greedy partition is near-tight on realistic data.

Method / Feedback noise	Temporal	Open Domain	Multi-Hop	Single-Hop	Overall	Split Precision	Split Frequency
DeMem, $p = 0.0$	91.9	86.8	84.7	93.2	91.1	85.0	4.6
DeMem, $p = 0.1$	90.4	85.1	82.8	92.5	90.2	80.8	4.9
DeMem, $p = 0.2$	88.7	83.6	81.2	92.1	88.9	77.6	5.4
Mnemis	87.8	79.3	79.8	93.8	88.8	–	–
EMem-G	71.7	51.7	70.2	78.2	73.7	–	–
RAG	57.2	59.0	54.3	69.8	63.7	–	–

Table 18: Reward-channel robustness on LoCoMo under corrupted binary feedback. Noise is applied only to the feedback rewards used by DeMem for memory updates and certified split decisions; final answer quality is evaluated with the unchanged held-out evaluation judge. Split precision is measured against gold support annotations for post-hoc audit only.

Consistency with the approximation bridge. The measured greedy gap is also consistent with the distortion decomposition of Appendix E.10. There, the total observed distortion is 0.086, of which the compression floor accounts for $\hat{\epsilon}^{\star}_{\infty}(K) = 0.035$. If the greedy gap were substantial, the compression term would visibly exceed the oracle-partition floor; instead, the two are indistinguishable within measurement noise, providing further evidence that $\epsilon_e \approx \epsilon^{\star}_{\infty}(K)$ at the operating budget.

Summary. Across both controlled and realistic settings, the certified cannot-link graphs arising in agent-memory applications are sparse enough for the greedy partition to match or closely approximate the information-theoretic optimum. The tightest synthetic budget ($K = 3$) serves as a stress-test regime; at the operating budgets used in all benchmark experiments ($K \geq 8$), the greedy partition matches the oracle criterion within measurement resolution.

E.16 Reward-Channel Robustness on LoCoMo

DeMem uses post-decision scalar feedback to update value estimates and certified split decisions. In the offline LoCoMo protocol, we instantiate this feedback channel with a held-out LLM judge that is disjoint from the final evaluation judge. To evaluate sensitivity to feedback quality, we corrupt only the feedback stream used for memory updates while keeping the final answer evaluation unchanged.

Concretely, let $R_t \in \{0, 1\}$ denote the binary feedback score assigned to the answer at step t. We construct a noisy feedback channel

$$\widetilde{R}_t = \begin{cases} 1 - R_t, & \text{with probability } p, \\ R_t, & \text{with probability } 1 - p, \end{cases}$$

where $p \in \{0, 0.1, 0.2\}$. The corrupted reward $\widetilde{R}_t$ is used only for estimating action values and certified decision conflicts. It is never used to score final answers. Final answer quality is always computed using the original held-out evaluation judge under the same answer-time memory budget as in the main experiments.

Table 18 reports the robustness results on LoCoMo. As the feedback channel becomes noisier, DeMem degrades smoothly rather than collapsing. Under moderate corruption ($p = 0.1$), DeMem remains above the strongest memory baseline in the main comparison. Even at $p = 0.2$, its performance remains competitive, while split precision decreases gradually and split frequency increases only mildly. These results suggest that certified splitting does not require perfectly clean feedback. The smooth degradation pattern is consistent with the confidence-threshold design of the split rule: noisier feedback reduces split precision gradually rather than causing abrupt failure.

E.17 Comparison with feedback-aware memory baselines

The main benchmark tables compare DeMem against a broad set of retrieval-based, static, and fixed-procedure memory systems under the same answer-time memory budget. To further evaluate DeMem in an adaptive-memory setting, we additionally compare against two methods whose memory policies are shaped by downstream task signals or memory actions: MemSkill and MemAct.

MemSkill learns a controller for selecting memory skills using downstream task signals and evolves the skill bank from hard cases; MemAct formulates memory management as policy actions and trains a context-curation policy with reinforcement learning. These methods provide complementary adaptive-memory baselines beyond retrieval-only systems.

Table 19: Results on LoCoMo with Qwen2.5-14B-Instruct backbone. Measured by LLM-as-a-judge binary accuracy.

Method	Temporal	Open Dom.	Multi-Hop	Single-Hop	Overall
MemSkill	0.423	0.438	0.461	0.492	0.466
MemAct	0.401	0.469	0.445	0.478	0.450
DeMem	**0.512**	**0.531**	**0.503**	**0.487**	**0.490**

Table 19 reports results on LoCoMo with a Qwen2.5-14B-Instruct backbone. DeMem obtains the best overall score and the strongest Temporal, Open-Domain, and Multi-Hop performance, while MemSkill is slightly better on Single-Hop questions. These results indicate that decision-aware certified refinement remains effective when compared with learning-based memory policies, not only with static retrieval pipelines.

E.18 Compute resources

Experiments using GPT-4o-mini and GPT-4.1-mini were run through the OpenAI API under deterministic decoding. These hosted-model experiments did not use local GPU resources. The open-weight Qwen2.5-14B-Instruct experiments were run on a machine with $2\times$ NVIDIA H100 80GB GPUs.

The reported results do not involve training new foundation models. GPU usage is limited to open-weight model inference, while GPT-based answering and judging are performed through hosted API calls.

F Scope and Limitations

Theory-to-practice correspondence. Our formal analysis isolates the answer-time memory bottleneck: the current query is observed directly, while the relevant history is accessed through one of K runtime memory states. This is the component controlled by the memory budget in our implemented agent. The deployed system instantiates the formal objects through routed dialogue histories, bounded memory slots, deterministic answers, and benchmark feedback.

Appendix D.8 provides the corresponding operational mapping for the encoder, memory state, policy, action, reward, distortion, and split relation. Appendix D.9 further decomposes realized distortion into compression, routing, and realization terms, and Appendix E.10 measures these terms empirically. The main ablations and mechanism audits then test the components singled out by this decomposition, including routing, splitting, memory representation, DEMEM-CORE, split precision, memory-state compatibility, and partition interventions (Figure 3(b); Appendices D.10, E.11, E.8, E.9).

These analyses are intended to make the abstraction operational: the theory specifies the decision bottleneck, while the implementation tests whether the corresponding routing, refinement, and slot-organization mechanisms account for the observed benchmark gains.

Scalability and feedback quality. The current implementation evaluates split evidence over pairs of distinct routing events observed within a dialogue. This is tractable at the benchmark scale considered here (Table 9). For substantially larger context vocabularies, the same split-and-certify structure can be combined with candidate generation, batching, locality-sensitive retrieval, or approximate nearest-neighbor search, so that split tests are evaluated on plausible conflict pairs rather than all pairs.

DeMem refines memory from scalar post-decision feedback, matching the online decision formulation while keeping the runtime memory budget fixed. In offline conversational benchmarks, this feedback channel is instantiated with held-out answer-level scorers that are separate from the final evaluation judges in Section 5. Appendix E.17 compares against feedback-aware and memory-action baselines, and Appendix E.16 evaluates corrupted feedback channels. The results show smooth degradation as feedback becomes noisier, indicating that the refinement mechanism relies on aggregate feedback reliability rather than perfectly clean observations.

G Broader Impact Statement

This work establishes that the right unit of analysis for agent memory is not information preserved, but decisions preserved. This distinction is not merely technical — it redraws the boundary between what an agent needs to know and what it can afford to forget, grounding that boundary in a quantity that is measurable, optimizable, and provably tight.

At a foundational level, our results contribute a missing piece to the theory of bounded rationality in language agents: a formal account of how finite memory should be allocated when the objective is action, not reconstruction. The rate-distortion frontier we derive is, to our knowledge, the first characterization of the fundamental limits of budgeted agent memory in terms of achievable decision quality rather than representational fidelity. The NP-completeness result (Theorem 3) further reveals that optimal memory allocation is computationally irreducible in the worst case — a structural insight that constrains the design space of all future memory systems, not only ours.

Practically, as language agents are deployed in high-stakes, long-horizon settings — clinical dialogue, legal case management, personalized education, multi-session negotiation — the question of *what to remember* becomes inseparable from the question of *what to do*. Systems that compress memory by descriptive salience risk retaining vivid but decision-irrelevant detail while discarding subtle but action-critical distinctions. Our framework provides a principled alternative: memory is judged by whether it preserves the capacity to act well, and forgetting is permitted exactly when it does not compromise that capacity. This criterion is architecture-agnostic (Appendix E.12) and can serve as a drop-in evaluation lens or selection module for existing memory systems.

Decision-centric memory optimizes what information is retained for downstream task performance. In deployed systems, this objective should be paired with application-specific retention policies, user-facing transparency, and periodic audits of stored and forgotten information. These safeguards are complementary to our technical framework and help align memory optimization with privacy, governance, and user-control requirements.

More broadly, we hope this work encourages the community to evaluate agent memory not by how much it remembers, but by whether it remembers what matters.

H Additional Evaluation of Cost, Reliability, and Backbones

H.1 End-to-End Inference and Memory-Maintenance Cost

We measure the complete LoCoMo evaluation trajectory with GPT-4o-mini. Accounting includes all method-internal LLM operations: memory construction and updates, retrieval or routing control, counterfactual candidate generation, feedback scoring, guarded conflict testing, and final answer generation. Token counts include input and output tokens. Initial construction is amortized over the downstream queries associated with each dialogue. The common final benchmark-evaluation judge is excluded for all methods. For Mnemis and EMem-G, the accounting likewise covers their indexing, graph construction, retrieval, filtering, and selection pipelines.

Table 20: End-to-end cost on LoCoMo with GPT-4o-mini. Tokens are thousands per query. Build includes construction, maintenance, candidate generation, feedback scoring, and conflict testing; control denotes query-time retrieval/control. Latency is end-to-end seconds per query.

Method	Calls	Build	Control	Answer	Total	Seconds	Score
Longer context	1.0	0.0	0.0	13.2	13.2	21.0	0.681
Mnemis	4.6	35.4	1.2	4.6	41.2	28.4	0.888
EMem-G	4.1	0.7	4.6	1.5	6.8	20.1	0.737
DeMem	4.0	3.8	0.0	5.8	9.6	24.8	0.911

DeMem uses 27.3% fewer amortized tokens than longer-context inference while requiring three additional calls and 3.8 s more latency per query. EMem-G consumes fewer tokens than DeMem but has a lower score in this setting. Thus, Table 20 describes a cost–quality tradeoff.

Matched total token budget. We also match the entire method-internal token allowance, rather than only the memory injected at answer time. Table 21 shows that the performance gap persists at 9.6k total tokens per query. Answer-time context tokens count the supplied context, whereas the answer column of Table 20 counts input and output of final answer generation.

Table 21: LoCoMo performance at a matched end-to-end token budget. Token counts are thousands per query.

Setting	Total tokens	Answer-time context	Overall
Longer context, matched total budget	9.6	9.4	0.674
DeMem, full accounting	9.6	5.6	0.911

One-off use and amortization. DeMem's construction cost makes one-off use more expensive (Table 22). The measured 25.2k-token initialization cost and approximately 9.4k incremental tokens per query give the token comparison $25.2 + 9.4n$ versus $13.2n$, in thousands, for n queries reusing the memory. The crossover is $n \geq 7$. This is a token-cost boundary: even the full-trajectory average retains a latency overhead. The 9.6k trajectory average and the approximate incremental-cost model are separately reported measurements.

Table 22: One-off and amortized cost on LoCoMo.

Setting	Total tokens/query	Seconds/query
Longer context, one-off	13.2k	21.0
DeMem, one-off	34.6k	71.4
DeMem, complete-trajectory average	9.6k	24.8

H.2 Paired Significance of the Main Comparisons

All methods answer the same histories and queries, so uncertainty is assessed using paired comparisons. We use 10,000 dialogue/session-level paired bootstrap resamples for 95% confidence intervals and paired permutation tests. Questions originating from the same dialogue or source history are resampled together to preserve their dependence. The eight predefined comparisons are DeMem versus Mnemis and EMem-G on LoCoMo and LongMemEval with GPT-4o-mini and GPT-4.1-mini. Reported permutation p-values are Holm-adjusted across these eight comparisons.

Table 23: Paired gains in overall score. Intervals are 95% paired bootstrap intervals; p_{H} is the Holm-adjusted permutation p-value.

Dataset / backbone	Comparator	Gain [95% CI]	p_{H}
LoCoMo / GPT-4o-mini	Mnemis	$0.023\ [0.003, 0.043]$	0.036
	EMem-G	$0.174\ [0.136, 0.213]$	0.012
LoCoMo / GPT-4.1-mini	Mnemis	$0.014\ [-0.007, 0.035]$	0.109
	EMem-G	$0.108\ [0.074, 0.143]$	0.012
LongMemEval / GPT-4o-mini	Mnemis	$0.018\ [0.003, 0.033]$	0.036
	EMem-G	$0.092\ [0.069, 0.115]$	< 0.001
LongMemEval / GPT-4.1-mini	Mnemis	$0.024\ [0.010, 0.039]$	0.016
	EMem-G	$0.066\ [0.045, 0.087]$	< 0.001

Seven comparisons remain significant at level 0.05 after correction. The LoCoMo/GPT-4.1-mini gain over Mnemis is positive in point estimate, but its interval overlaps zero. Consequently, the higher mean in that configuration should not be interpreted as a statistically established gain.

H.3 Generalization to Additional Answering Backbones

We compare Mnemis and DeMem on LongMemEval with Qwen3.5-9B, Qwen3.5-122B-A10B, and GPT-5.4 mini. Within each backbone comparison, both methods use the same benchmark split, prompts, decoding protocol, answering model, and answer-time memory budget. GPT-5.4 mini is evaluated with reasoning disabled; the two Qwen models use non-thinking mode. Qwen3.5-122B-A10B is an open-weight sparse-MoE model accessed through a hosted API, whereas Qwen3.5-9B is dense.

Table 24: Additional backbone comparisons on LongMemEval. Gain is DeMem minus Mnemis in overall score.

Answering backbone	Mnemis	DeMem	Gain
Qwen3.5-9B	0.852	0.870	$+0.018$
Qwen3.5-122B-A10B	0.899	0.922	$+0.023$
GPT-5.4 mini	0.914	0.931	$+0.017$

The gains range from 1.7 to 2.3 percentage points, extending the observed benefit across these model scales and architectures. These results provide point-estimate evidence of backbone generalization; the paired tests in Appendix H.2 apply to the original two backbones.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's theoretical, algorithmic, and empirical contributions.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses scope and limitations in Appendix F, including the theory–system bridge, scalability of conflict testing, and dependence on reward signal quality. Appendix H.1 reports full inference costs and the memory-reuse boundary.

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]

   Justification: The paper states the assumptions of the contextual decision model explicitly, including the finite context/action spaces, the i.i.d. context distribution, and the hard runtime memory budget.

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]

   Justification: The paper specifies the synthetic environment, benchmark preprocessing, answer-time memory-budget matching protocol, prompting and judge setup, ablation variants, hyperparameter sweeps, and supplementary mechanism audits in the main text and appendices. Together, these descriptions provide a concrete path to reproducing the reported results, up to the use of third-party closed-source API backbones in some evaluations.

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: [Yes]

   Justification: The supplemental material includes an anonymized implementation of the proposed DeMem method, together with prompts, configuration files, evaluation scripts, and instructions for reproducing the main DeMem experiments under the protocol described in the paper. Public datasets and third-party baselines are referenced through their official releases. A complete public repository with additional scripts and documentation will be released after the anonymous review period, subject to policy and third-party license constraints.

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]

   Justification: The experimental setup, benchmark protocol, memory-budget matching, ablations, and supplementary audits are described in the main paper and appendices.

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We report uncertainty estimates for the synthetic, diagnostic, and benchmark analyses. Unless otherwise stated, tabular benchmark results are reported as mean $\pm$ one sample standard deviation over independent full-pipeline runs.

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]

   Justification: Appendix E.18 reports the available compute-resource information: GPT-4o-mini and GPT-4.1-mini were used through the OpenAI API, and Qwen2.5-14B-Instruct experiments were run on 2$\times$ NVIDIA H100 80GB GPUs.

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]

   Justification: We have reviewed the NeurIPS Code of Ethics and, to the best of our knowledge, the research reported in this paper conforms to it. We also preserve anonymity in the submission and discuss relevant methodological and societal considerations in the paper and appendix.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses both positive and negative societal impacts of improved long-horizon agent memory in Appendix G.

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a high-risk generative model, a scraped dataset, or another asset that would require special release safeguards. The work studies memory organization and evaluates it on existing benchmarks and third-party model backbones.

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]

    Justification: The paper cites the original papers and official releases for the datasets, models, and baseline systems used in the experiments. Appendix E.2.2 summarizes the existing assets used for evaluation and comparison, and notes that their use follows the corresponding release terms or provider policies. No third-party assets are redistributed as part of this submission.

13. **New assets**

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

    Answer: [N/A]

Justification: The submission does not release new public assets at submission time. Although the paper includes additional annotations and audits for analysis, these are used internally for evaluation rather than introduced as a released public asset in the current submission.

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

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

    Answer: [Yes]

    Justification: Appendix E.14.1 describes the human annotation protocol used for the LoCoMo judge-agreement study, including annotator qualifications, task setup, the full instructions provided to annotators, recorded information, compensation, consent procedure, risk assessment, and use of the annotations. The study was used only to validate the LLM-as-a-judge protocol and was not used to train DeMem, tune hyperparameters, select models, or construct the reported benchmark test set.

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: [Yes]

    Justification: Appendix E.14.1 describes the potential risks, consent procedure, information recorded from annotators, and ethics-review status of the human annotation study. The annotation protocol was reviewed under the authors' institutional procedures and determined to be exempt minimal-risk evaluation research; institution-identifying details are omitted during anonymous 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]

    Justification: LLMs are an important experimental component of this work: we evaluate the proposed memory system with LLM answering backbones and use held-out LLM judges for answer assessment.