What Is Agent Memory? Definition, Storage, and Performance Prediction

Anonymous authors
Paper under double-blind review

Abstract

Memory (the capacity to store and recall information over time) is essential for large language model (LLM) and reinforcement learning (RL) agents solving complex, partially observable tasks. While AI agent design widely adopts the cognitive taxonomy of episodic memory (isolated events) and semantic memory (synthesized general knowledge), existing analyses primarily describe *what content* each memory type holds rather than how they differ structurally or when a given mechanism will succeed. We close this gap by proposing an information-theoretic and computational theory of episodic and semantic memory for AI agents acting in POMDPs. Measuring efficiency via cell-probe time and space complexity and defining deployment success as maximizing expected reward under finite compute and memory budgets, we prove that episodic and semantic tasks impose opposing structural constraints: simultaneously achieving fast $O(1)$ isolated updates/lookups for episodic tasks and minimal $O(H(A))$ compressed space for semantic tasks is mathematically impossible. Applying this framework, we classify six popular memory mechanisms (spanning vector RAG, graph memory, and long-context transformers) and evaluate them across 32 controlled diagnostic tasks to show how structural mismatches trigger accuracy collapse or out-of-memory failures. Finally, a meta-analysis of five existing benchmarks confirms that structural alignment between the memory mechanism and task explains a median of 11% of performance variance.

1 Introduction

Demand for large language model (LLM) and reinforcement learning (RL) agents to complete increasingly complex tasks is growing (Sumers et al., 2024; Zhao et al., 2023; Silver et al., 2017). Memory is widely recognized as necessary for these capabilities because complex tasks often require reasoning over imperfect information, massive data storage, or intermediate caching (Hu et al., 2026; Sumers et al., 2024). Yet, our current understanding of AI agent memory is limited to descriptive taxonomies of memory content and existing mechanisms (Cherepanov et al., 2024; Hu et al., 2026; Du et al., 2025). While useful, descriptive taxonomies do not help us assess whether a memory mechanism's structural properties (e.g., its time and space complexity) are well-suited for a given task. This makes predicting deployment accuracy and resource consumption difficult (Cherepanov et al., 2024). Confounding the problem further, many existing evaluations collapse memory into a single metric or test multiple regimes unsystematically, producing misleading results and obscuring the contributions of specific memory mechanisms (Lee et al., 2025; Bian et al., 2026).

To address this gap, we take conceptual inspiration from how psychology (Goldstein, 2011) and neuroscience (O'Reilly et al., 2012) distinguish between episodic and semantic memory (Hu et al., 2026). In neuroscience, these are recognized not merely by their content, but as fundamentally different information structures designed to avoid distinct cognitive failure modes such as catastrophic forgetting between overlapping events and failure to generalize across related experiences. While our work draws intuition from this biological dichotomy, our core contribution is strictly computational: we provide an information-theoretic categorization and complexity analysis of partially observable Markov decision processes (POMDPs),

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

establishing a general framework to analyze the space and time complexities of any artificial memory mechanism (see section 4).

Structurally, biological episodic memories are stored in the hippocampus as isolated events, where interactions between discrete memories are minimized to prevent interference (O'Reilly et al., 2012), analogous to a hash map (O'Reilly et al., 2012, Secs. 7.2 and 7.3). Conversely, semantic memories are consolidated in the neocortex and rely on rich relational links between synthesized facts to improve compressibility and utility, requiring interconnected structures like graphs or neural networks rather than isolated lookup tables (O'Reilly et al., 2012, Sec. 7.3). Consequently, episodic storage seeks to minimize interactions between stored items to support fast, isolated updates and retrieval, whereas semantic storage seeks to maximize interactions to compress redundant structure.

Building on this structural distinction, we formalize episodic and semantic memory as distinct computational regimes within a POMDP and prove that a single memory representation cannot be universally optimal for both simultaneously (theorem 3). Specifically, achieving expected $O(1)$ update and lookup times requires encoding variables independently, which directly precludes the joint encoding needed to achieve minimal $O(H(A))$ space complexity (theorem 2). Moreover, an agent cannot trivially bypass this trade-off by maintaining both memory types in parallel without cost: duplicating all observations across both an isolated store and a compressed store forfeits $O(H(A))$ space optimality (scaling instead as $O(\sum_{i=1}^{n} H(A_i))$) unless paired with a non-trivial routing mechanism capable of separating independent facts from relational structure online.

We further demonstrate why these theoretical complexity bounds directly predict task performance under realistic deployment constraints (lemma 6). Because deployed agents operate under finite computational and context budgets, a mechanism whose operational cost scales super-optimally for a task will eventually exceed its budget as the horizon grows, forcing incomplete retrieval or state truncation and strictly bounding achievable reward. We formalize this across episodic and semantic POMDP regimes, noting that while this binary partition is simply one useful way of categorizing POMDPs, the underlying complexity trade-offs (e.g., the incompatibility of $O(1)$ updates and $O(H(A))$ space) and our analytical framework remain invariant. Within this categorization, we prove that: (1) near-optimal efficiency on one episodic task transfers across all episodic tasks (theorem 4), (2) near-optimal efficiency on an episodic task implies sub-optimal space efficiency on semantic tasks and vice versa (theorem 2), and (3) near-optimal efficiency on one semantic task does not guarantee transferability to semantic tasks governed by different relational structures (theorem 5).

We empirically validate these theoretical predictions in controlled settings by evaluating six popular memory mechanisms (spanning RAG retrieval, long-context attention, context-window management, and graph-based memory) across 32 synthetic tasks in 8 task families. We design these controlled tasks to isolate pure multi-information and query regimes without pretraining or domain confounders. Consistent with our theory, episodic mechanisms (e.g., RAG, HippoRAG) degrade sharply on semantic tasks requiring relational synthesis, while semantic mechanisms face scaling and out-of-memory bottlenecks on long-horizon episodic tasks unless aided by targeted compression. Furthermore, validating theorem 5, no single semantic architecture dominates across all semantic relational structures (e.g., sequential state tracking vs. graph topology).

Finally, to demonstrate that our theoretical insights generalize beyond simple controlled settings and provide practical predictive value, we conduct a meta-analysis across five existing real-world memory benchmarks. We show that correctly matching episodic and semantic memory mechanisms to tasks explains a median of 11% of the variance in performance across existing benchmarks.

2 Related Work

Our work bridges theoretical reinforcement learning (RL) memory formulations with empirical large language model (LLM) agent architectures. By linking information-theoretic

abstractions to storage constraints, we provide a unified framework for predicting memory performance.

Theoretical Formulations of Memory In RL, memory is frequently formalized as a necessary component for solving partially observable Markov decision processes (POMDPs). A prominent theoretical perspective frames memory as an abstraction over the environment's state space or observed trajectories (Li et al., 2006; Brafman & De Giacomo, 2024; Kirtland et al., 2025). These works focus on the desirable properties of the abstraction function itself. However, they generally do not address the architectural constraints required to actually store and update these abstractions. We adopt the view of memory as an abstraction of observation history, but shift the focus from the agent's ability to *create* the abstraction to the computational properties (time and space complexity) required to *store* it efficiently.

LLM Agent Memory Architectures In contrast to RL theory, research into LLM agent memory is highly empirical and mechanism-driven (Zhao et al., 2023). Current literature primarily focuses on systematizing existing methodologies or providing descriptive taxonomies limited strictly to language modalities (Liang et al., 2025; Du et al., 2025; Wu et al., 2026; Yang et al., 2026). Furthermore, many proposed frameworks rely heavily on biomimicry (He et al., 2025) or are explicitly "chatbot-centric," assuming conversational structures like user preferences (Hu et al., 2026; Huang et al., 2026). Closest to our analytical approach is Dorovatas et al. (2026), who evaluate theoretical properties like update speeds for parametric versus non-parametric memory. While we also distinguish parametric from stateful (non-parametric) memory, our analysis operates orthogonally by characterizing how stateful memory itself can be structured across episodic and semantic regimes. Rather than treating these two regimes as an exhaustive partition of all memory, we use them to establish foundational space and time complexity limits that apply to any artificial memory mechanism.

Mathematical Psychology Our information-theoretic approach draws conceptual inspiration from models of biological memory, notably Gershman (2021) and Nagy et al. (2025), who frame human memory as a rate-distortion problem by treating observation histories as inputs to a limited-capacity channel. However, those works stop at introducing the general rate-distortion framing for biological cognition. Adapting this perspective to artificial agents acting in POMDPs is non-trivial and requires formalizing how stored variables $A_1, \ldots, A_n$ act as sufficient statistics coupled with explicit data-structure operations and query distributions (see section 3.3). Both our application of this framing to AI agent architectures and our core theoretical insights are novel.

An extended discussion of related work is provided in appendix B.

3 A Definition of Memory for AI Agents

3.1 Preliminaries

Problem Setup. We represent AI agents as acting within a partially observable Markov decision process (POMDP) (Sondik, 1971) defined by the 7-tuple $(S, \mathcal{A}_{\text{act}}, T, R, \Omega, O, \gamma)$, where S is the state space, $\mathcal{A}_{\text{act}}$ is the action space, T defines conditional state transition probabilities, $R : S \times \mathcal{A}_{\text{act}} \to \mathbb{R}$ is the reward function, Ω is the observation space, O provides conditional observation probabilities, and $\gamma \in [0, 1]$ is a discount factor. A memoryless agent learns a policy $\pi(a \mid o)$ over current observation $o \in \Omega$ to maximize the value function $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, seeking an optimal policy π^* such that $V^{\pi^*}(s) \geq V^\pi(s)$ for all $s \in S$.

Computational Model. We evaluate theoretical time and space complexity in the cell-probe model (Yao, 1981), where memory is partitioned into cells of word size $w \geq \log n$ (n total stored elements). Operational time cost is measured strictly by the number of memory cell accesses (reads and writes), treating internal computations on accessed words as free.

This yields unconditional lower bounds on data-structure efficiency independent of hardware implementation.

3.2 Defining Memory Storage Locations

Motivating Example. Consider the classical T-maze problem (Olton, 1977), where an agent begins at the bottom of a "T"-shaped hallway and must walk to either the left or right top arm based on a binary cue $x \in \{0, 1\}$ visible exclusively at $t = 0$. While trivial if the agent observes the full state $s \in S$, a memoryless policy $\pi(a_t \mid o_t)$ fails when $o_t \in \Omega$ reveals only current corridor coordinates: upon reaching the top junction, o_t contains only position and no information about x.

Solving this POMDP requires extending the agent with an external, updatable m-bit memory vector $M \in \mathbb{Z}_2^m$ managed by a memory mechanism $\mathcal{M}$. We formally define $\mathcal{M}$ as a unified transition and readout function taking current memory state $M_t \in \mathbb{Z}_2^m$ and an operation $u \in \mathcal{U}$ (either a write $u(o_t)$ triggered by $o_t \in \Omega$ or a read query), returning the updated state and an optional readout from $\mathcal{Y}$:

$$\mathcal{M} : \mathbb{Z}_2^m \times \mathcal{U} \to \mathbb{Z}_2^m \times (\mathcal{Y} \cup \{\emptyset\}).$$

For a write operation, $(M_{t+1}, \emptyset) = \mathcal{M}(M_t, u(o_t))$. In the T-maze, $m = 1$ bit suffices: $\mathcal{M}$ writes $M_1 = x$ at $t = 0$, and the memory-augmented policy $\pi(a_t \mid o_t, M_t)$ queries M_t at the junction to recover x, matching full-state return. Analytically separating π from $\mathcal{M}$ isolates structural storage constraints from policy reasoning.

Definition 1 (Memory Solvable)**.** Given a task formulated as a POMDP $(S, \mathcal{A}_{\text{act}}, T, R, \Omega, O, \gamma)$, the task is *memory solvable* if there exists a finite bit-length $m \in \mathbb{N}$, a memory vector $M \in \mathbb{Z}_2^m$, a mechanism $\mathcal{M} : \mathbb{Z}_2^m \times \mathcal{U} \to \mathbb{Z}_2^m \times (\mathcal{Y} \cup \{\emptyset\})$, and a memory-augmented policy $\pi^*_M(a \mid o, M)$ achieving the value of an optimal fully observable policy $\pi^*_s(a \mid s)$ across all states: $V^{\pi^*_s}(s) = V^{\pi^*_M}(s)$ for all $s \in S$.

Definition 2 (Stateful Memory)**.** Given a memory-solvable task, the *stateful memory* required to solve the task is the minimum bit-length $m \in \mathbb{N}$ such that there exists $M \in \mathbb{Z}_2^m$ rendering the task memory solvable (Brafman & De Giacomo, 2024).

Beyond the $m = 1$ bit of trajectory-specific stateful memory in M_t, an agent must also statically encode environment dynamics, atomic skills, and the parameters governing π and $\mathcal{M}$ learned prior to deployment. We distinguish this static knowledge as *parametric memory*.

Definition 3 (Parametric Memory)**.** Given an agent parameterized by θ executing policy π_θ and mechanism $\mathcal{M}_\theta$, the *parametric memory* of the agent is the static, episode-invariant information encoded within the fixed parameters θ prior to deployment, independent of the active trajectory's stateful memory vector $M_t \in \mathbb{Z}_2^m$.

3.3 Defining a Memory Task

We define a *memory task* as any POMDP task requiring at least $m \geq 1$ bit of stateful memory; otherwise, all task-relevant information can reside in static parametric memory. Over an episode $O_{1:T} = (o_1, \ldots, o_T)$, the agent extracts n task-relevant elements $E = \{(k_1, A_1), \ldots, (k_n, A_n)\}$, where each $k_i \in \mathcal{K}$ is an exogenous identifier (e.g., timestep, entity name, or item ID) and each $A_i \in \mathcal{A}$ is a random variable drawn from the trajectory-induced joint distribution P_A over $A = (A_1, \ldots, A_n)$.

For the agent to act optimally, the extracted variables $A = f(O_{1:T})$ must be a deterministic abstraction of the trajectory ($H(A \mid O_{1:T}) = 0$) forming a sufficient statistic of $O_{1:T}$ with respect to target query outputs Y, satisfying $I(A; Y) = I(O_{1:T}; Y)$ (or $I(O_{1:T}; Y \mid A) = 0$). Expanding $H(O_{1:T}, A)$ and applying sufficiency ($H(O_{1:T} \mid A) = H(O_{1:T} \mid A, Y)$) yields the exact relationship between trajectory entropy and stored memory entropy:

$$H(O_{1:T}, A) = H(O_{1:T}) + H(A \mid O_{1:T}) = H(A) + H(O_{1:T} \mid A) \implies H(A) = H(O_{1:T}) - H(O_{1:T} \mid A, Y). \tag{1}$$

Here, $H(O_{1:T} \mid A, Y) \geq 0$ quantifies task-irrelevant environmental noise discarded during storage. When $H(O_{1:T}) \gg H(A)$, an efficient mechanism must heavily filter or compress the trajectory into M.

Task and Query Distributions. To maximize reward, the agent executes operations $u \in \mathcal{U}$ from a task-induced distribution $P_{\mathcal{U}}$ spanning writes and reads. Restricting $P_{\mathcal{U}}$ to the read subset $\mathcal{Q} \subset \mathcal{U}$ defines the conditional query distribution $P_Q(q) = P_{\mathcal{U}}(u = q \mid u \in \mathcal{Q})$. A query $q : \mathcal{X} \to \mathcal{Y}$ maps input $x \in \mathcal{X}$ to $q(x) \in \mathcal{Y}$ without mutating M, i.e., $\mathcal{M}(M, (q, x)) = (M, q(x))$. We focus on two primary classes:

- **Lookup Queries ($\mathcal{Q}_{\text{lookup}}$):** Retrieve a single stored value by its exogenous key ($\mathcal{X} = \mathcal{K}$) independent of other stored elements: $q_{\text{lookup}}(k^*) = A_i$ if and only if $(k^*, A_i) \in E$.
- **Relational Queries ($\mathcal{Q}_{\text{relational}}$):** Evaluate structural dependencies across the unordered value set $\mathbf{A}_{\text{set}} = \{A_i \mid (k_i, A_i) \in E\}$ via a predicate $\Phi : \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A}) \to \{\text{True}, \text{False}\}$, extracting all qualifying subsets $q_{\text{relational}}(\Phi) = \big\{S \subseteq E \;\big|\; \Phi(\{A_i \mid (k_i, A_i) \in S\}, \mathbf{A}_{\text{set}}) = \text{True}\big\}$.

3.4 Defining Episodic and Semantic Tasks

Memory mechanisms must structure the compression of $O_{1:T}$ into $A = (A_1, \ldots, A_n)$ according to inter-variable dependencies and P_Q (figs. 2 and 3). Episodic tasks require recalling isolated, self-contained facts with near-zero mutual information, whereas semantic tasks require synthesizing interdependent facts via a joint encoding.

Definition 4 (Episodic Task)**.** Given an information tolerance $\epsilon > 0$ and a query threshold $\delta \in (0, 1)$, a task over n variables is (ϵ, δ)-episodic if the stored variables exhibit near-zero multi-information, $I(A_1, \ldots, A_n) = \sum_{i=1}^{n} H(A_i) - H(A_1, \ldots, A_n) \leq \epsilon$, and P_Q overwhelmingly favors lookup queries: $P_{q \sim P_Q}(q \in \mathcal{Q}_{\text{lookup}}) \geq 1 - \delta$.

Definition 5 (Semantic Task)**.** Given a compression factor $\epsilon \in (0, 1)$ and a query threshold $\delta \in (0, 1)$, a task over n variables is (ϵ, δ)-semantic if the joint entropy is bounded by a fraction of the marginal entropies, $H(A_1, \ldots, A_n) \leq \epsilon \sum_{i=1}^{n} H(A_i)$ (equivalently, $I(A_1, \ldots, A_n) \geq (1 - \epsilon) \sum_{i=1}^{n} H(A_i)$), and P_Q overwhelmingly favors relational queries: $P_{q \sim P_Q}(q \in \mathcal{Q}_{\text{relational}}) \geq 1 - \delta$.

Absolute vs. Relative Information Bounds. Note the intentional asymmetry between the two definitions: definition 4 bounds multi-information by an *absolute* constant $\epsilon > 0$, as independent facts remain uncorrelated even as n scales. Conversely, definition 5 bounds joint entropy by a *relative* factor $\epsilon \in (0, 1)$ of $\sum_{i=1}^{n} H(A_i)$, ensuring that as total entropy grows with n, at least a $(1 - \epsilon)$ proportion of marginal entropy is shared structure that a joint encoding can asymptotically compress.

3.5 Task-Specific Memory Mechanisms

Because episodic and semantic tasks impose opposing structural demands on storage and retrieval, we categorize memory mechanisms by how they organize $M \in \mathbb{Z}_2^m$.

Definition 6 (Episodic Mechanism)**.** An episodic memory mechanism $\mathcal{M}_{\text{ep}}$ maps exogenous identifiers $k_i \in \mathcal{K}$ to isolated segments of $M \in \mathbb{Z}_2^m$ such that total space complexity scales with the sum of marginal entropies, $\Theta\big(\sum_{i=1}^{n} H(A_i)\big)$ bits, while achieving expected $O(1)$ cell-probe time complexity for lookup, insertion, and deletion operations.

Definition 7 (Semantic Mechanism)**.** A semantic memory mechanism $\mathcal{M}_{\text{sem}}$ maintains a joint encoding over $M \in \mathbb{Z}_2^m$ to exploit inter-variable redundancy, achieving sub-additive space complexity $O(H(A_1, \ldots, A_n))$ bits (strictly smaller than $\sum_{i=1}^{n} H(A_i)$ whenever $I(A_1, \ldots, A_n) > 0$), while indexing structural dependencies to answer a target family of relational queries $\mathcal{Q}_{\text{relational}}(\Phi)$ in $o(n)$ cell probes (strictly faster than an exhaustive $\Omega(n)$ scan of isolated records).

4 Properties of Efficient Memory Mechanisms

Predominant AI agent memory architectures such as vector RAG or monolithic long-context windows are frequently deployed as "one-size-fits-all" general-purpose stores (Zhao et al., 2023; Wu et al., 2026; Hu et al., 2026). Evaluating complexity limits in the cell-probe

model (Yao, 1981), we prove that no single memory mechanism can be universally optimal across episodic and semantic regimes. Moreover, naively running both stores in parallel without an online router duplicates all observations into the $\Theta(\sum_{i=1}^{n} H(A_i))$ episodic store, forfeiting $O(H(A))$ space optimality whenever $I(A_1, \ldots, A_n) > 0$.

4.1 Space–Time Complexity Trade-offs

Lemma 1 (Asymptotic Space Optimality)**.** *For a sequence of random variables $A = (A_1, \ldots, A_n)$ from a stationary ergodic source, a memory mechanism can achieve optimal $O(H(A_1, \ldots, A_n)/w)$ cell-probe space complexity for word width w without prior knowledge of the source distribution.*

Proof Sketch & Significance. Universal prefix-free coding (e.g., Lempel-Ziv (Cover & Thomas, 2012)) converges asymptotically to the entropy rate H_∞, packing A into $\lceil (H(A) + o(n))/w \rceil$ cells (appendix C). *Significance:* This establishes that ideal semantic mechanisms can compress redundant trajectories down to their true joint entropy $O(H(A))$ without knowing environment statistics in advance but only by constructing a tightly coupled joint encoding.

Theorem 2 (Incompatibility of Optimal Space and Constant-Time Operations)**.** *Given variables $A = \{A_1, \ldots, A_n\}$ with growing multi-information $I(A_1; \ldots; A_n) = \omega(w)$, any mechanism maintaining A within optimal $O(H(A))$ bits ($O(H(A)/w)$ cells) cannot support expected $O(1)$-time insertion of a dependent variable A_{n+1} ($I(A_{n+1}; A) > 0$), $O(1)$-time in-place updates, or $O(1)$-time isolated key lookup.*

Proof Sketch & Significance. By the pigeonhole principle, compressing $\sum_{i=1}^{n} H(A_i)/w$ cells into $H(A)/w$ cells forces at least $k = I(A_1; \ldots; A_n)/w = \omega(1)$ memory cells to be shared across coupled variables. Reconstructing this conditioning context to insert A_{n+1} at rate $H(A_{n+1} \mid A)$, decode an isolated A_i, or re-encode coupled states upon an in-place update requires probing $\Omega(k) = \omega(1)$ shared cells (appendix D). *Significance:* This is our foundational architectural trade-off: a mechanism engineered for semantic space compression ($O(H(A))$) must sacrifice the $O(1)$ isolated insertions, updates, and lookups required by episodic tasks (and vice versa).

4.2 Impossibility of Universal Operational Optimality

Definition 8 (Universally Optimal Mechanism)**.** For a task set $\mathcal{T}$, let $C^*_\tau(u)$ be the optimal cell-probe time complexity for operation $u \in \mathcal{U}$ by a mechanism specialized solely for $\tau \in \mathcal{T}$. A mechanism $\mathcal{M}^*$ is *universally optimal* across $\mathcal{T}$ if it achieves $O(C^*_\tau(u))$ for all $\tau \in \mathcal{T}$ and $u \in \mathcal{U}$.

Theorem 3 (Impossibility of Universal Operational Optimality)**.** *There does not exist a single memory mechanism $\mathcal{M}^*$ that is universally optimal across both (ϵ, δ)-episodic and (ϵ, δ)-semantic tasks simultaneously.*

Proof Sketch & Significance. Matching the standalone optimal $O(1)$ dynamic `insert`/`delete` of an episodic hash table while simultaneously matching the $O(1)$ `find_smallest` query of a static ordered semantic pointer reduces to implementing a dynamic priority queue, which unconditionally requires $\Omega(\log \log n)$ probes per operation (Thorup, 2000; Beame & Fich, 2002) (appendix E). *Significance:* Even with unlimited memory space, opposing query structures prevent any single data structure from achieving optimal latency across both episodic and semantic tasks.

4.3 Transferability Across Tasks

Theorem 4 (Universal Transferability of Episodic Mechanisms)**.** *Let $\mathcal{M}_{ep}$ be an episodic mechanism (definition 6) capable of hashing any key domain $\mathcal{K}$ to isolated segments with worst-case non-lookup time $T_{\max}(n)$. If $\mathcal{M}_{ep}$ is optimal on one (ϵ, δ)-episodic task, it maintains optimal $O(1)$ expected operation time and $O(\sum_{i=1}^{n} H(A_i))$ space across all (ϵ, δ)-episodic tasks satisfying $\delta = O(1/T_{\max}(n))$.*

Proof Sketch & Significance. Because $I(A_1, \ldots, A_n) \leq \epsilon$, independent storage is space-optimal across any episodic domain, and expected query cost $(1-\delta)O(1) + \delta T_{\max}(n)$ remains $O(1)$ whenever $\delta = O(1/T_{\max}(n))$ (appendix F). *Significance:* Episodic memory is "plug-and-play": a single well-tuned episodic store (e.g., key-value or vector lookup) transfers near-optimally across arbitrary episodic tasks without structural redesign.

Conversely, let a semantic mechanism be *standalone-optimal* for predicate Φ if its update (t_u) and query (t_q) times match a structure dedicated solely to Φ.

Theorem 5 (Limits of Semantic Transferability)**.** *No single semantic mechanism $\mathcal{M}_{sem}$ can simultaneously achieve standalone-optimal update and query time complexities across all relational query structures $\mathcal{Q}_{relational}(\Phi)$.*

Proof Sketch & Significance. Simultaneously matching $t_u = O(1)$ for $R_1 =$ FIND_LAST_INSERTED and $t_s = \Theta(\log n)$ for $R_2 =$ FIND_KTH_SMALLEST violates Ptracu and Demaine's (Pătraşcu & Demaine, 2005) cell-probe bound $t_u \log(t_s/t_u) = \Omega(\log n)$, forcing a polynomial query blowup $t_s = n^{\Omega(1)}$ (appendix G). *Significance:* Unlike episodic memory, semantic memory is not universally transferable even within the semantic regime: a representation structured for sequential state tracking will degrade on graph or order-statistic reasoning.

4.4 Performance Under Resource Constraints

Lemma 6 (Efficiency Bounds Performance Transfer)**.** *For a memory-solvable task $\mathcal{T}$ with optimal cost $C^*(n)$ and operational budget $B(n) \leq c \cdot C^*(n)$ ($c \geq 1$), any structurally mismatched mechanism $\mathcal{M}$ with $Cost(\mathcal{M}, \mathcal{T}, n) = \omega(C^*(n))$ exceeds $B(n)$ for all $n > N$, strictly bounding expected return: $V^{\pi_{\mathcal{M}}}(s) < V^{\pi_s^*}(s)$ for at least one $s \in S$.*

Proof Sketch & Significance. For $n > N$, budget exhaustion halts $\mathcal{M}$ prematurely, truncating $A_{\text{truncated}}$ so it violates the sufficient-statistic condition $I(A_{\text{truncated}}; Y) < I(O_{1:T}; Y)$ and forces suboptimal policy actions (appendix H). *Significance:* This bridges asymptotic complexity to empirical accuracy: under finite context windows, retrieval limits (k), or VRAM budgets, architectural mismatch directly causes accuracy collapse or out-of-memory failure once task scale crosses N.

5 Experimental Setup

To isolate memory regimes without benchmark confounders, we introduce a controlled diagnostic suite of 8 task families across 4 difficulty scales (L1–L4, 32 configurations) and evaluate 6 memory mechanisms using a fixed **Qwen3-4B** backbone.[1] Fixing the backbone θ is without loss of generality because our bounds govern the external stateful mechanism $\mathcal{M}$ (definition 3), while our meta-analysis (section 7) verifies generalization across diverse LLM families. Following our POMDP formalism (section 3), environment messages act as observations $o_t \in \Omega$ during a dynamic *memory formation phase* ($u(o_t) \in \mathcal{U}$), followed by a *frozen-memory probe phase* where M_T is locked and the agent answers queries $q \sim P_Q$.

Diagnostic Tasks and Mechanisms. Full specifications and classifications appear in appendices I.1 to I.4. Our **Episodic Tasks (definition 4)** (*Name-Attributes*, *Graph Edge Presence*, and *Inventory Ledger (Snapshot)*) store independent facts ($I(A_1, \ldots, A_n) \leq \epsilon$) for lookup queries ($\mathcal{Q}_{\text{lookup}}$). Our **Semantic Tasks (definition 5)** (*Stack*, *Queue*, *Graph 1-Hop Neighborhood Sum*, *Graph Shortest Path Distance*, and *Inventory Ledger (Delta)*) require synthesizing interdependent streams for relational queries ($\mathcal{Q}_{\text{relational}}$). We compare **Episodic Mechanisms (definition 6)** (*RAG* and *HippoRAG* (Gutiérrez et al., 2024) which store isolated chunks/triples for $O(1)$ top-k retrieval) against **Semantic Mechanisms (definition 7)** *Everything-in-Context (EiC)*, *Attention Sink* (Xiao et al., 2023), *Specialized-in-Context* (iterative LLM state compression), and *HippoRAG with Attributes* (jointly binding node values as structural graph attributes).

[1] Code and configurations: `https://anonymous.4open.science/r/MemoryExp-8B6F`.

6 Results

6.1 Episodic Mechanisms Fail to Generalize to Semantic Tasks

Consistent with theorem 4, episodic mechanisms (*RAG*, *HippoRAG*) transfer reliably across episodic lookup tasks (fig. 1b). However, as predicted by theorems 2 and 3, their isolated storage ($I(A_1; \ldots; A_n) \approx 0$) causes severe collapse on semantic tasks (fig. 1a). On sequential tasks (*Stack*, *Queue*, *Inventory Ledger Delta*), $O(1)$ top-k retrieval truncates intermediate operations, violating sufficiency ($I(A_{\text{truncated}}; Y) < I(O_{1:T}; Y)$) and driving accuracy toward zero as horizon length scales (lemma 6). Likewise, localized queries fail to capture multi-hop graph dependencies (*Neighborhood Sum*, *Shortest Path*). Comparing semantic mechanisms further validates theorem 5: context-based mechanisms (*EiC*, *Specialized-in-Context*) excel on sequential FIFO/LIFO tasks but degrade on *Shortest Path*, whereas *HippoRAG with Attributes* improves relational graph tracking, confirming that no single semantic store is optimal across all relational structures.

6.2 Semantic Mechanisms Exhibit Budget Violations and Scaling Trade-offs on Episodic Tasks

As shown in fig. 1b, semantic mechanisms exhibit scale-dependent accuracy on episodic tasks governed by the budget threshold N (lemma 6) and our VRAM measurements (fig. 4 in appendix J). At small scales (L1–L2, $n < N$), the full trajectory fits within the context budget $B(n)$, allowing *EiC* and *Attention Sink* to brute-force attend over all tokens. However, at L3–L4 ($n > N$), the space–time cost of uncompressed joint storage (theorem 2) triggers severe hardware bottlenecks (fig. 4): whereas *RAG*, *HippoRAG*, and *Specialized-in-Context* maintain flat $O(1)$ active VRAM ($\sim$800–1,200 MiB), *EiC*'s peak VRAM grows super-linearly from $\sim$1 GiB (L1) to $>$ 30 GiB (L4), degrading *Name-Attributes* recall and causing fatal out-of-memory (OOM) failures ($Cost > B(n)$) at L4 on *Inventory Ledger*. *Attention Sink* caps generation cache ($\sim$300 MiB at L1–L3) but still spikes to $>$ 20 GiB during full-transcript prefill at L4, while *Specialized-in-Context* avoids VRAM explosion only by paying an $\Omega(k)$ state-rewriting time cost at every update step (theorem 2).

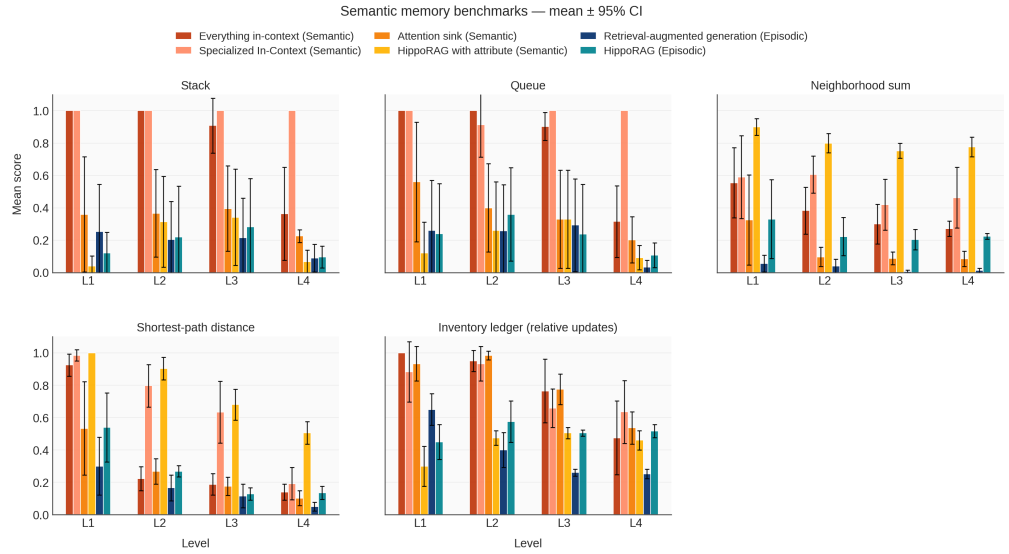
7 Meta-Analysis: Structural Alignment Predicts Benchmark Performance

To verify that our theoretical framework generalizes beyond controlled synthetic settings and across diverse LLM backbones, we conduct a meta-analysis of published results across five established memory benchmarks: Episodic Memory Eval (Huet et al., 2025), LoCoMo (Maharana et al., 2024), LongMemEval_S (Wu et al., 2024), MemBench (Tan et al., 2025), and Memory AgentBench (Hu et al., 2025) (methodology in appendix K; sample data in table 3). We classify each evaluated algorithm and benchmark task as episodic or semantic according to definitions 4 to 7 and test whether structural alignment (matched vs. mismatched mechanism–task pairs) predicts normalized score distributions using a one-way ANOVA (table 1).

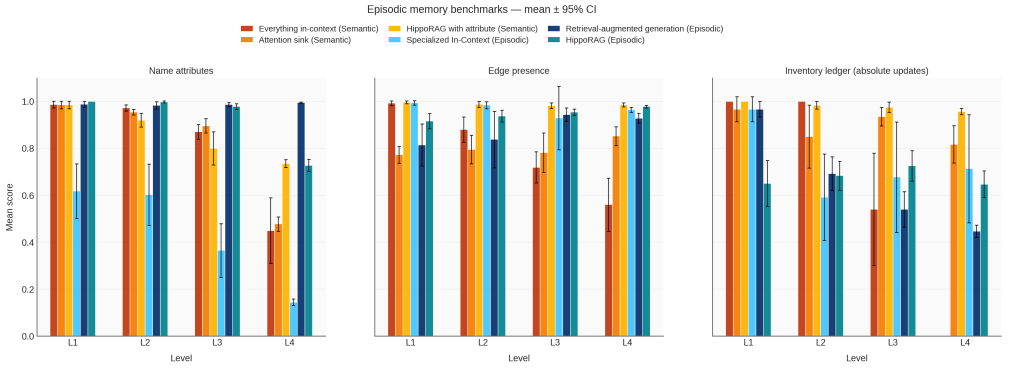
Four of the five benchmarks yield statistically significant alignment effects ($p < 0.05$), with mechanism–task matching explaining a **median of 11% of total performance variance** ($\eta^2 = 0.1077$ on Episodic Memory Eval and $\eta^2 = 0.1154$ on LoCoMo, up to $\eta^2 = 0.6869$ on LongMemEval_S, $p < 10^{-4}$). The sole null result (Memory AgentBench, $p = 0.8563, \eta^2 = 0.0006$) is directly explained by its aggregation of conflicting episodic and semantic sub-tasks into a single composite score, which masks regime-specific advantages (full analysis in appendix K.1).

8 Discussion

We formalized episodic and semantic memory for POMDP agents, proving that $O(1)$ isolated operations and $O(H(A))$ joint compression are mutually incompatible (theorems 2 and 3), characterizing intra-regime transferability (theorems 4 and 5), and demonstrating that



(a) Semantic Benchmarks



(b) Episodic Benchmarks

Figure 1: Mean accuracy across diagnostic benchmarks for all six agents across difficulty scales L1–L4 (95% CIs; orange: semantic mechanisms; blue: episodic mechanisms). **(a)** Semantic mechanisms outperform episodic stores on relational tasks, with structure-specific variation (theorem 5). **(b)** On episodic tasks, semantic mechanisms match baselines at small scales ($n < N$) but suffer degradation or OOM budget violations (*EiC* at L4) as scale grows (lemma 6; VRAM scaling in fig. 4).

Table 1: Per-benchmark one-way ANOVA results testing the effect of mechanism–task alignment on benchmark performance. $\eta^2 = \mathrm{SS}_{\mathrm{between}}/\mathrm{SS}_{\mathrm{total}}$ denotes the proportion of variance explained (median $\eta^2 = 10.77\% \approx 11\%$).

Benchmark	F	ANOVA p	Statistically Significant	η^2
Episodic Memory Eval	4.5879	0.0387	Yes	0.1077
LoCoMo	5.9982	0.0182	Yes	0.1154
LongMemEval_S	39.4908	< 0.0001	Yes	0.6869
MemBench	5.7855	0.0040	Yes	0.0860
Memory AgentBench	0.0331	0.8563	No	0.0006

structural alignment explains both synthetic scaling bottlenecks and a median of 11% of variance across existing benchmarks. Notably, because our framework isolates storage constraints, this analysis to the full memory lifecycle (including deciding what information to store and how to retrieve information efficiently) is a compelling direction for future work.

AI use statement

In this work, we used generative AI tools for:

- To aid or polish writing: The first couple paper drafts were made humans. Most polishing was done by LLM tools
- Retrieval and discovery (e.g., finding related work): We used LLM research tools to identify seminal papers in the domains the we (the authors) were unfamiliar with. After identifying seminal papers we looked at papers citing these seminal works to identify the remaining literature. We also used LLM research tools to help us find survey papers.

We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results. And the rest of the required disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for:

- generating plots
- summarize existing literature
- edit a research paper to improve readability

We have reviewed all AI-assisted work. All LLM generated plotting code was manually verified and test to be correct. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

To ensure full reproducibility of both our theoretical and empirical contributions, we provide complete formal proofs and explicit assumptions for all lemmas and theorems in appendices C to H. Complete descriptions of our 8 diagnostic task families, scaling parameters across difficulty levels L1–L4, random seeds, agent hyperparameters, and exact evaluation metrics are detailed in appendices I.1 to I.6, alongside hardware and compute specifications in appendix J. Full source code, experiment configuration files, run scripts, and the raw meta-analysis dataset (also summarized in appendix K and table 3) are publicly available in our anonymous repository at `https://anonymous.4open.science/r/MemoryExp-8B6F`.

References

Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning of pomdps using spectral methods, 2016. URL `https://arxiv.org/abs/1602.07764`.

Paul Beame and Faith E. Fich. Optimal bounds for the predecessor problem and related problems. *Journal of Computer and System Sciences*, 65(1):38–72, 2002. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.2002.1822. URL `https://www.sciencedirect.com/science/article/pii/S0022000002918222`.

Haonan Bian, Zhiyuan Yao, Sen Hu, Zishan Xu, Shaolei Zhang, Yifu Guo, Ziliang Yang, Xueran Han, Huacan Wang, and Ronghao Chen. Realmem: Benchmarking llms in real-world memory-driven interaction, 2026. URL `https://arxiv.org/abs/2601.06966`.

Ronen I. Brafman and Giuseppe De Giacomo. Regular decision processes. *Artificial Intelligence*, 331:104113, 2024. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2024.104113. URL `https://www.sciencedirect.com/science/article/pii/S0004370224000493`.

Egor Cherepanov, Nikita Kachaev, Artem Zholus, Alexey K. Kovalev, and Aleksandr I. Panov. Unraveling the complexity of memory in rl agents: an approach for classification and evaluation, 2024. URL https://arxiv.org/abs/2412.06531.

T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2012. ISBN 9781118585771. URL https://books.google.com/books?id=VWq5GG6ycxMC.

Vaggelis Dorovatas, Malte Schwerin, Andrew D. Bagdanov, Lucas Caccia, Antonio Carta, Laurent Charlin, Barbara Hammer, Tyler L. Hayes, Timm Hess, Christopher Kanan, Dhireesha Kudithipudi, Xialei Liu, Vincenzo Lomonaco, Jorge Mendez-Mendez, Darshan Patil, Ameya Prabhu, Elisa Ricci, Tinne Tuytelaars, Gido M. van de Ven, Liyuan Wang, Joost van de Weijer, Jonghyun Choi, Martin Mundt, and Rahaf Aljundi. Modular memory is the key to continual learning agents, 2026. URL https://arxiv.org/abs/2603.01761.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in llm based agents: Representations, operations, and emerging topics, 2025. URL https://arxiv.org/abs/2505.00675.

Samuel J Gershman. The rational analysis of memory. *Oxford handbook of human memory.*, 2021.

Samuel J. Gershman and Nathaniel D. Daw. Reinforcement learning and episodic memory in humans and animals: An integrative framework. *Annual Review of Psychology*, 68: 101–128, 2017. doi: 10.1146/annurev-psych-122414-033625.

Samuel J. Gershman, Ila Fiete, and Kazuki Irie. Key-value memory in the brain, 2025. URL https://arxiv.org/abs/2501.02950.

E. Bruce Goldstein. *Cognitive psychology: Connecting mind, research, and everyday experience: CogLab Manual*. Wadsworth Cengage Learning, 2011.

Zhaohan Daniel Guo, Shayan Doroudi, and Emma Brunskill. A pac rl algorithm for episodic pomdps, 2016. URL https://arxiv.org/abs/1605.08062.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W. Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. Human-inspired perspectives: A survey on ai long-term memory, 2025. URL https://arxiv.org/abs/2411.00489.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025.

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.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed A. Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Wang, Yongfeng Zhang, Jiaxuan You, Hanghang

Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. Rethinking memory mechanisms of foundation agents in the second half: A survey, 2026. URL https://arxiv.org/abs/2602.06052.

Alexis Huet, Zied Ben Houidi, and Dario Rossi. Episodic memories generation and evaluation benchmark for large language models. *International Conference on Learning Representations*, 2025.

Chi Jin, Sham M. Kakade, Akshay Krishnamurthy, and Qinghua Liu. Sample-efficient reinforcement learning of undercomplete pomdps, 2020. URL https://arxiv.org/abs/2006.12484.

Aaron Kirtland, Alexander Ivanov, Cameron Allen, Michael L. Littman, and George Konidaris. Memory as state abstraction over trajectories. In *6th Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 2025.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. Realtalk: A 21-day real-world dataset for long-term conversation, 2025. URL https://arxiv.org/abs/2502.13270.

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.

Jiafeng Liang, Hao Li, Chang Li, Jiaqi Zhou, Shixin Jiang, Zekun Wang, Changkai Ji, Zhihao Zhu, Runxuan Liu, Tao Ren, Jinlan Fu, See-Kiong Ng, Xia Liang, Ming Liu, and Bing Qin. Ai meets brain: Memory systems from cognitive neuroscience to autonomous agents, 2025. URL https://arxiv.org/abs/2512.23343.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of llm agent memory mechanisms. *Preprints*, January 2026. doi: 10.20944/preprints202601.0618.v2. URL https://doi.org/10.20944/preprints202601.0618.v2.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

David G. Nagy, Gergő Orbán, and Charley M. Wu. Adaptive compression as a unifying framework for episodic and semantic memory. *Nature Reviews Psychology*, 4(7):484–498, July 2025. ISSN 2731-0574. doi: 10.1038/s44159-025-00458-6. URL https://doi.org/10.1038/s44159-025-00458-6.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment, 2023. URL https://arxiv.org/abs/2307.03864.

David S Olton. Spatial memory. *Scientific American*, 236(6):82–99, 1977.

Randall C O'Reilly, Yuko Munakata, Michael J Frank, Thomas E Hazy, et al. Computational cognitive neuroscience. 2012.

Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *CoRR*, abs/cs/0502041, 2005. URL http://arxiv.org/abs/cs/0502041.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Edward Jay Sondik. *The optimal control of partially observable Markov processes*. Stanford University, 1971.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents, 2024. URL https://arxiv.org/abs/2309.02427.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.989. URL https://aclanthology.org/2025.findings-acl.989/.

Zhenheng Tang, Xin He, Tiancheng Zhao, Fanjunduo Wei, Xiang Liu, Peijie Dong, Qian Wang, Qi Li, Huacan Wang, Ronghao Chen, Sen Hu, Weidong Guo, Yu Xu, Haolan Chen, Kunfeng Lai, Kaiyong Zhao, Keyan Ding, Ivor W. Tsang, Yew-Soon Ong, Bo Li, and Xiaowen Chu. Llm agent memory: A survey from a unified representationmanagement perspective. *Preprints*, March 2026. doi: 10.20944/preprints202603.0359.v2. URL https://doi.org/10.20944/preprints202603.0359.v2.

Mikkel Thorup. On ram priority queues. *SIAM Journal on Computing*, 30(1):86–109, 2000. doi: 10.1137/S0097539795288246. URL https://doi.org/10.1137/S0097539795288246.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. 2024. URL https://arxiv.org/abs/2410.10813.

Yanchen Wu, Tenghui Lin, Yingli Zhou, Fangyuan Zhang, Qintian Guo, Xun Zhou, Sibo Wang, Xilin Liu, Yuchi Ma, and Yixiang Fang. Memory in the llm era: Modular architectures and strategies in a unified framework, 2026. URL https://arxiv.org/abs/2604.01707.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv*, 2023.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. Graph-based agent memory: Taxonomy, techniques, and applications, 2026. URL https://arxiv.org/abs/2602.05665.

Andrew Chi-Chih Yao. Should tables be sorted? *J. ACM*, 28(3):615628, July 1981. ISSN 0004-5411. doi: 10.1145/322261.322274. URL https://doi.org/10.1145/322261.322274.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

A Supplementary Figures

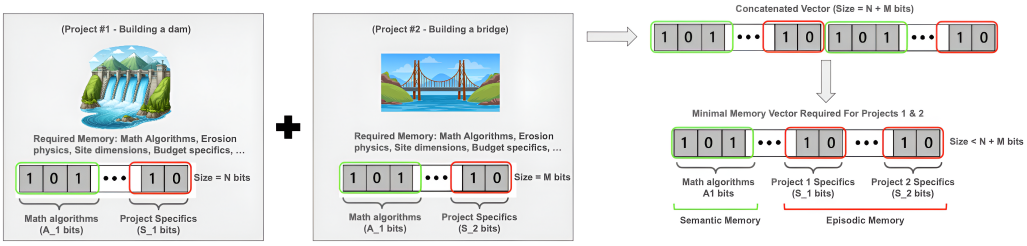


Figure 2: Semantic and episodic memory require different storage structures. Shared knowledge (math algorithms, erosion physics) has high mutual information and can be compressed below $A_1 + A_2$ bits via joint encoding. Project-specific facts (site dimensions, budget specifics) are independent and must be stored separately, proportional to the sum of individual entropies. Efficient semantic storage exploits inter-variable dependencies; efficient episodic storage isolates variables to support $O(1)$ lookup.

Memory Systems: Specifics vs. Generalities

Category	Episodic Task	Semantic Task
Goals:	Remember **Specifics**	Extract **Generalities**
Example / Query:	Where is the car parked today? lookup("today's spot")	What is the best parking strategy? best_spot(D1, D2, D3, ...)
Information:	**Low Mutual Information**	**High Mutual Information**
Solution:	**Separate** Reps D1 D2 D3 ... D1 D2 D3 ...	**Overlapping** Reps PS (parking strategy) D1 D2 D3 ...
Learning Speed:	**Fast learning** (encode immediately)	**Slow learning** (integrate over many observations)
	Learn automatically (encode everything)	**Task-driven learning** (extract relevant information)
Storage Structure:	[0] Day 1 → Spot A3 [1] Day 2 → Spot B7 [2] Day 3 → Spot C1 [3] Day 4 → Spot A9	

Figure 3: Episodic and semantic tasks require fundamentally different query types. In episodic tasks, stored facts are independent, meaning relational queries are uninformative. The only meaningful operation is a direct lookup. In semantic tasks, stored variables are highly interdependent, enabling and requiring relational queries that reason across multiple entries.

Figures 2 and 3 provide intuition about semantic and episodic tasks.

B Extended Related Work

Our work relates to three primary fields: RL theory, LLM agent memory design, and mathematical psychology. Each of these fields attempts to define and explain memory in different ways.

RL Theory of Memory Many works in RL theory focus on categorizing classes of non-Markovian Decision Processes (MDPs), such as partially observable MDPs (POMDPs), based on the efficiency of known learning algorithms.

For handling uncertainty in POMDPs, Sondik (1971) introduced belief states, which are distributions over all states in the environment representing the likelihood an agent is in each of the states given the agent's observation history. Belief states neatly describe the uncertainty about the environment's true underlying state, and an agent's ability to accurately maintain a belief state depends on its memory ability. However, belief states are difficult to work with when reasoning about an agent's memory mechanisms because they do not provide any insight into how an agent ought to store information. It is also difficult to use them to answer questions such as "How many bits of memory are required to solve this task?"

Another way to think about memory is as an abstraction. Li et al. (2006) frame memory as an abstraction over the state space of the environment. They think of the abstraction as a function from the environment's state space to a new state space, and they consider desirable properties of this state abstraction function—for example, ensuring the actions taken by some policy are the same under the abstraction, or that the optimal policy remains the same in the abstract state space. Kirtland et al. (2025) view memory as an abstraction over trajectories. They build on the earlier work of Brafman & De Giacomo (2024), who view memory as the state of a finite state machine that updates its state based on observations and actions from the environment. Kirtland et al. (2025) consider the types of abstractions this finite state machine implicitly induces on its observed trajectory, i.e., how it maps a trajectory to one of its internal states. We also think of memory as an abstraction of previous observations. However, unlike the above works, we do not focus on an agent's ability to create these abstractions. Rather, we consider the categories of abstractions an agent will need to create (i.e., depending on the amount of mutual information an agent must store) and define properties of an efficient architecture to store such abstractions.

Ni et al. (2023) analyze memory by considering *memory length* and *credit assignment length*. They define multiple memory lengths, each describing the minimum number of past items (e.g., observations, actions) required from the full history to achieve optimal performance. They define the credit assignment length of a policy within an environment as the maximum number of time steps between an action and its associated reward signal. These definitions study memory by looking at the number of time steps for which information must be stored. While useful, these definitions give limited insight into the mechanisms required to store information, which is our topic of interest.

POMDPs are a popular formalism for environments with incomplete information requiring memory. Jin et al. (2020); Azizzadenesheli et al. (2016); Guo et al. (2016) approach the analysis of memory by considering the existence of efficient learning algorithms within various subclasses of POMDPs. Thus, one can think of their work as implicitly categorizing memory via a categorization of environments that require memory. While this categorization is useful, it is currently unclear how to leverage it to improve our understanding of memory storage architectures.

LLM Agent Memory Most works studying memory design for LLM agents approach the problem empirically (Zhao et al., 2023). Nearly all works defining and categorizing memory for LLM agents limit their definitions strictly to LLM agents, i.e., RL agents cannot be categorized using these definitions (Liang et al., 2025; Hu et al., 2026; Du et al., 2025; Wu et al., 2026; Huang et al., 2026; Luo et al., 2026; Tang et al., 2026; Yang et al., 2026). While useful when studying LLM agents, our goal is to establish definitions that categorize all agents. Additionally, many of these memory categorizations only systematize existing methodologies

for memory mechanisms rather than considering the full set of possible mechanisms (Liang et al., 2025; Hu et al., 2026; Wu et al., 2026; Huang et al., 2026; Luo et al., 2026). Some of these definitions are also "chatbot-centric" (Hu et al., 2026; Huang et al., 2026), including user preferences as core components of memory even though many agents do not have a notion of user preferences. While valuable, these works differ from ours in that we make claims about the fundamental properties of any agent's memory mechanism, including agents across all modalities (e.g., LLMs, RL) and memory mechanisms yet to be discovered.

He et al. (2025) is not limited to LLM agents, and if one considers replacing each of the components they define (e.g., the sensory register, working memory, long-term memory adapter, etc.) with arbitrary implementations, this definition can be quite general. For example, agents with no sensory memory can be represented by considering the sensory register to only store the current observation. However, many of the components they introduce (e.g., the long-term memory adapter component) are closely tied to how human memory works, making it difficult to represent memory systems that differ substantially from human cognition.

Dorovatas et al. (2026) examine parametric and non-parametric memory and compare properties such as update speed for each. This is similar to our analysis of properties for efficient memory mechanisms. However, we analyze episodic and semantic regimes within stateful (non-parametric) memory and use these properties to draw conclusions about efficient storage structures.

Mathematical Psychology This field attempts to explain psychological mechanisms, including memory, within animals by describing them mathematically. Gershman (2021) and Nagy et al. (2025) frame memory as a rate-distortion theory problem. They think of observations as inputs to a channel with limited capacity. The observations are then encoded into memory, which they equate to the transmitted signal. Later, the signal (i.e., the memory) is decoded (i.e., read), and the output (i.e., the information necessary to solve the task) is extracted. This is analogous to how we think about stateful memory. Our definition of stateful memory (definition 2) is very close to what they call "the signal." We build on this work by extending the idea to the machine learning setting, where, unlike for animal brains, notions of pre-training and offline learning exist. We do this by introducing parametric memory (definition 3), which stores information learned outside of the current trajectory.

Gershman & Daw (2017) define episodic memory based on the conjecture that it is used to store representative trajectory subsets, which are then used to dynamically estimate the state value function. This fits nicely with our work, since these representative trajectory subsets are likely to have low mutual information between one another. However, their work differs in that they propose a specific purpose for episodic memory, whereas we characterize the computational constraints for storing this information.

Gershman et al. (2025) consider how animal brains and neural networks can perform key-value storage. This relates to our architectural constraints on storing episodic information, but focuses more on the biological and neural mechanism for key-value retrieval rather than fundamental cell-probe complexity limits.

C Proof of Lemma 1

Proof. Apply Lempel-Ziv coding to the observed sequence $A_1, \ldots, A_n$ to produce a prefix-free encoding of length $L(A_1, \ldots, A_n)$. By the asymptotic optimality of Lempel-Ziv on stationary ergodic sources (Cover & Thomas, 2012), the per-symbol code length converges to the source entropy rate H_∞:

$$\frac{L(A_1, \ldots, A_n)}{n} \to H_\infty \quad \text{as } n \to \infty.$$

For stationary sources, $H(A_1, \ldots, A_n) = nH_\infty + o(n)$, so $L(A_1, \ldots, A_n) = H(A_1, \ldots, A_n) + o(n)$. Packing the bitstream into cells of width w requires

$$S(n) = \left\lceil \frac{L(A_1, \ldots, A_n)}{w} \right\rceil = \frac{H(A_1, \ldots, A_n) + o(n)}{w}.$$

To evaluate the asymptotic efficiency, we divide the actual space $S(n)$ by the theoretical minimum $H(A_1, \ldots, A_n)/w$:

$$\frac{S(n)}{H(A_1, \ldots, A_n)/w} = \frac{\frac{H(A_1,\ldots,A_n)+o(n)}{w}}{\frac{H(A_1,\ldots,A_n)}{w}} = 1 + \frac{o(n)}{H(A_1, \ldots, A_n)}.$$

Since $H(A_1, \ldots, A_n) = nH_\infty + o(n)$ grows linearly with n, and $o(n)$ by definition grows strictly slower than linear time, the fractional term $\frac{o(n)}{H(A_1,\ldots,A_n)}$ approaches 0 as $n \to \infty$.

Therefore, as $n \to \infty$, $S(n)/(H(A_1, \ldots, A_n)/w) \to 1$, so Lempel-Ziv asymptotically achieves the information-theoretic optimum without prior distribution knowledge. □

D Proof of Theorem 2

Proof. Consider a set of existing variables $A = \{A_1, \ldots, A_n\}$ with growing multi-information $I(A_1; \ldots; A_n) = \omega(w)$ stored in memory within the optimal space bound of $O(H(A)/w)$ cells, where w is the cell width.

Independent storage of the variables would require $\sum_{i=1}^{n} H(A_i)/w$ cells, whereas the optimal joint encoding requires only $H(A)/w$ cells. By the pigeonhole principle, achieving this optimal compression means that at least

$$k = \sum_{i=1}^{n} \frac{H(A_i)}{w} - \frac{H(A)}{w} = \frac{I(A_1; \ldots; A_n)}{w} = \omega(1)$$

memory cells must be "shared" across coupled variables to encode their mutual information.

1. Impossibility of $O(1)$ Insertion. When a dependent variable A_{n+1} with $I(A_{n+1}; A) > 0$ is inserted, maintaining the optimal $O(H(A, A_{n+1})/w)$ space bound dictates that A_{n+1} must be encoded incrementally at its conditional entropy rate, occupying $O(H(A_{n+1} \mid A)/w)$ additional cells. Because $I(A_{n+1}; A) > 0$, the exact optimal bit-representation of A_{n+1} mathematically depends on the specific realized values of the coupled variables in A. Under the cell-probe model, the memory mechanism $\mathcal{M}$ evaluates operations statelessly based only on the current memory vector. To compute the correct conditional encoding for A_{n+1}, $\mathcal{M}$ must dynamically reconstruct the conditioning context by reading the shared cells encoding the mutual information of A. Because $k = \omega(1)$ grows without bound as n increases, each insertion requires $\Omega(k) = \omega(1)$ cell probes, precluding $O(1)$ insertion time.

2. Impossibility of $O(1)$ Isolated Key Lookup and In-Place Updates. Similarly, for an isolated key lookup $\mathcal{Q}_{\text{lookup}}(k_i)$ or an in-place update to an existing variable A_i, because $I(A_1; \ldots; A_n) = \omega(w)$, the information content of A_i is jointly compressed across the $k = \omega(1)$ shared memory cells rather than isolated in a constant number of dedicated cells. Decoding A_i upon lookup requires reading $\Omega(k) = \omega(1)$ shared cells to resolve the joint conditioning context. Likewise, modifying A_i in-place while preserving the optimal $O(H(A)/w)$ joint encoding requires re-encoding the coupled state and writing to $\Omega(k) = \omega(1)$ shared cells. Consequently, a mechanism maintaining optimal $O(H(A))$ space cannot support expected $O(1)$-time insertion, $O(1)$-time in-place updates, or $O(1)$-time isolated key lookup. □

E Proof of Theorem 3

Proof. We proceed by contradiction. Assume there exists a universally optimal memory mechanism $\mathcal{M}^*$ capable of serving both episodic and semantic tasks optimally.

Consider an episodic task (definition 4) requiring the storage of an arbitrary mapping from an index set to stored variables. The optimal memory mechanism for this task (e.g., a dynamic hash table with universal hashing) achieves expected $O(1)$ time complexity for `insert`, `delete`, and `lookup`. By definition 8, $\mathcal{M}^*$ must support dynamic `insert` and `delete` operations in expected $O(1)$ time.

For ordered domains, we can instantiate the `find_smallest` relational query by restricting the output of the query to singleton sets $S = \{(k_i, A_i)\}$ and defining the predicate as $\Phi(\{x\}, A) = \forall y \in A, (x \leq y)$.

Next, consider a semantic task (definition 5) requiring the extraction of ordered relations. The optimal static memory mechanism for answering only a `find_smallest` query (e.g., maintaining a direct pointer to the minimum element) achieves $O(1)$ time complexity. By definition 8, $\mathcal{M}^*$ must support the `find_smallest` relational query in $O(1)$ time.

Because $\mathcal{M}^*$ is universally optimal, it must simultaneously support expected $O(1)$ `insert`, $O(1)$ `delete`, and $O(1)$ `find_smallest` operations over a dynamic sequence of elements.

However, a data structure that simultaneously supports dynamic insertions, deletions, and minimum extractions is functionally equivalent to a dynamic priority queue. It is a proven lower bound that no memory system can maintain a continuously ordered state in $O(1)$ time across all operations. In the cell-probe model, the inherent computational bottleneck of tracking ordered relations while continuously adding or removing elements requires at least $\Omega(\log \log n)$ memory accesses per operation (Thorup, 2000; Beame & Fich, 2002).

The theoretical requirement that $\mathcal{M}^*$ operates entirely in $O(1)$ memory accesses contradicts these unconditional cell-probe lower bounds. Therefore, the assumption that a universally optimal memory mechanism $\mathcal{M}^*$ can simultaneously achieve the lower bounds of both episodic and semantic memory architectures is false. □

F Proof of Theorem 4

Proof. Let $\mathcal{T}_1$ be the initial episodic task where $\mathcal{M}_{\text{ep}}$ is known to be optimal, and let $\mathcal{T}_2$ be any other arbitrary (ϵ, δ)-episodic task satisfying $\delta = O(1/T_{\max}(n))$. We must show that $\mathcal{M}_{\text{ep}}$ is also optimal for $\mathcal{T}_2$.

By the definition of an (ϵ, δ)-episodic task (definition 4), $\mathcal{T}_2$ requires the storage of variables $A_1, \ldots, A_n$ that exhibit near-zero multi-information, such that $I(A_1, \ldots, A_n) \leq \epsilon$. Because these variables are structurally isolated, they admit independent storage.

By our primary assumption (definition 6), $\mathcal{M}_{\text{ep}}$ is capable of mapping any arbitrary index set $\mathcal{K}$ to isolated memory segments. Therefore, regardless of the specific domain, format, or distribution of the exogenous keys in $\mathcal{T}_2$, $\mathcal{M}_{\text{ep}}$ can independently address and store the variables. Consequently, $\mathcal{M}_{\text{ep}}$ preserves its expected $O(1)$ time complexity for lookup, insertion, and deletion operations, as well as its optimal space complexity of $O(\sum_{i=1}^{n} H(A_i))$.

To prove time efficiency during deployment, we evaluate the expected query cost. By the definition of an episodic task, the query distribution P_Q for $\mathcal{T}_2$ overwhelmingly favors lookup queries, satisfying $P_{q \sim P_Q}(q \in \mathcal{Q}_{\text{lookup}}) \geq 1 - \delta$.

Let $T_{\max}(n)$ represent the worst-case time complexity of any non-lookup operation supported by $\mathcal{M}_{\text{ep}}$. The expected operational cost on $\mathcal{T}_2$ is strictly bounded by:

$$Cost(\mathcal{M}_{\text{ep}}, \mathcal{T}_2) \leq (1 - \delta) \cdot O(1) + \delta \cdot T_{\max}(n).$$

Substituting the condition $\delta = O(1/T_{\max}(n))$ into the bound yields:

$$Cost(\mathcal{M}_{\text{ep}}, \mathcal{T}_2) \leq (1 - \delta) \cdot O(1) + O\left(\frac{1}{T_{\max}(n)}\right) \cdot T_{\max}(n) = O(1).$$

Because $\mathcal{M}_{\text{ep}}$ maintains both expected $O(1)$ operation time and $O(\sum_{i=1}^{n} H(A_i))$ space complexity on $\mathcal{T}_2$, and because $\mathcal{T}_2$ was chosen as an arbitrary (ϵ, δ)-episodic task satisfying $\delta = O(1/T_{\max}(n))$, we conclude that $\mathcal{M}_{\text{ep}}$ transfers optimally across all such episodic tasks. □

G Proof of Theorem 5

Proof. We proceed by contradiction. Assume there exists a universally optimal semantic mechanism, $\mathcal{M}_{\text{sem}}$, capable of simultaneously achieving standalone-optimal time complexities for any distinct semantic relations.

Consider the relation $R_1 = \text{FIND_LAST_INSERTED}$. By maintaining a simple pointer, the optimal standalone data structure achieves an update time of $t_u = O(1)$ and a query time of $t_q^{(1)} = O(1)$.

Consider a second relation $R_2 = \text{FIND_KTH_SMALLEST}$. In the cell-probe model with word size $w = \Theta(\log n)$, an order-statistic tree achieves a tight standalone query time of $t_s = \Theta(\log n)$.

If $\mathcal{M}_{\text{sem}}$ is universally optimal across relational predicates, a single update to its underlying joint encoding must execute in $t_u = O(1)$ to satisfy R_1, while supporting R_2 queries in $t_s = \Theta(\log n)$. However, Pătraşcu & Demaine (2005) established unconditional cell-probe lower bounds for the dynamic `select` operation. Because $t_u \leq t_s$, $\mathcal{M}_{\text{sem}}$ must adhere to the symmetric form of their query–update trade-off:

$$t_u \cdot \log\left(\frac{t_s}{t_u}\right) = \Omega(\log n).$$

Crucially, this bound applies even if $\mathcal{M}_{\text{sem}}$ relies on amortized or randomized operations. Substituting the required $t_u = O(1)$ constraint from R_1 into the trade-off yields $O(1)\cdot\log(t_s) = \Omega(\log n)$. Solving for t_s reveals that the query time is strictly bounded by:

$$t_s = n^{\Omega(1)}.$$

This creates a definitive mathematical contradiction: maintaining an $O(1)$ update time forces the R_2 query time to undergo a polynomial blowup ($n^{\Omega(1)}$), making the required $\Theta(\log n)$ query time impossible. Therefore, no single data structure can simultaneously achieve the standalone optimal complexities for both R_1 and R_2, proving that semantic mechanisms cannot efficiently transfer across all relational structures. □

H Proof of Lemma 6

Proof. Let $\mathcal{T}$ be a memory-solvable task (definition 1) structured as a POMDP, requiring the storage of n variables, with an optimal operational complexity $C^*(n)$. We are given a dynamic computational or memory budget bounded by a constant factor of the optimum, $B(n) \leq c \cdot C^*(n)$ for $c \geq 1$.

Assume the agent utilizes a mismatched memory mechanism $\mathcal{M}$ such that its cost strictly dominates the optimal cost, i.e., $Cost(\mathcal{M}, \mathcal{T}, n) = \omega(C^*(n))$.

By the formal definition of little-omega (ω) asymptotic dominance, for any strictly positive constant c, there exists a threshold $N \in \mathbb{N}$ such that for all $n > N$:

$$Cost(\mathcal{M}, \mathcal{T}, n) > c \cdot C^*(n) \geq B(n).$$

Therefore, for any task progression where $n > N$, the required cell probes or memory cells to execute the operation $u \in \mathcal{U}$ strictly exceed the maximum allowable budget $B(n)$. Under the cell-probe model constraint, the mechanism $\mathcal{M}$ is forced to halt or truncate before completing its required read or write operation.

Let $A = (A_1, \ldots, A_n)$ be the stored variables. As established in section 3.3, optimal performance requires A to be a sufficient statistic of the observation history with respect to the query answers Y, satisfying $I(A; Y) = I(O_{1:T}; Y)$. When the operation halts prematurely due to the budget violation, the memory update or retrieval is truncated. This causes strict information loss, meaning the extracted response $y \in \mathcal{Y}$ or updated state M no longer preserves the sufficient statistic, yielding $I(A_{\text{truncated}}; Y) < I(O_{1:T}; Y)$.

Because $\mathcal{T}$ requires this information to resolve state uncertainty in the underlying POMDP (definition 1), the agent's memory-augmented policy $\pi_{\mathcal{M}}(a \mid o, M)$ is forced to act on incomplete or corrupted information. Therefore, there exists at least one true environment state $s \in S$ where the lack of sufficient information causes $\pi_{\mathcal{M}}$ to assign a higher probability to a sub-optimal action compared to the fully observable optimal policy π_s^*.

Since the policy deviates from optimal action selection, the expected discounted reward from state s must strictly decrease. Thus, we conclude:

$$V^{\pi_{\mathcal{M}}}(s) < V^{\pi_s^*}(s)$$

for at least one $s \in S$. □

I Experiment Details

I.1 Task Descriptions

Name-Attributes Task evaluates episodic memory. A set of named entities is sampled from a name pool, and each entity is assigned a subset of attributes drawn from eight field types: age, location, height (cm), occupation, hobby, phone number, employee ID, and account number. During the memory formation phase, attribute facts are provided to the agent one at a time in natural language (e.g., "Alice is 34 years old." or "Bob lives in Toronto."). The agent acknowledges each fact and forms relevant memories. At probe time, the agent receives a question in natural language at each turn of the form "Where does Bob live?" and must return the correct value from memory. No new facts are provided during the probe phase and the agent cannot update its memory buffer. Difficulty scales with the number of entities, the number of ingested facts, and the number of probe questions.

Inventory Ledger Task evaluates memory under sequential state updates. A set of items is initialized with a known default state (quantity 0 and status `active`), which is communicated to the agent at the start of the memory formation phase. Updates are then delivered sequentially as event observations of the form "`Event 7/30 [event_id=7]: ADD_STOCK item=I014 delta=11`". Three operation types are applied in random order: `add_stock` increments an item's quantity by the given delta; `remove_stock` decrements an item's quantity by the given delta; and `set_status` assigns a new categorical token drawn from a configurable pool (`active`, `hold`, or `discontinued`). The event ID increases monotonically at each turn. We evaluate two observation regimes (fixed per run): in *delta mode* (semantic), each turn states only the operation and its arguments, and the agent must integrate the stream into running totals; in *snapshot mode* (episodic), each turn also includes an authoritative snapshot of the affected item's current quantity and status after the update.

Stack Simulation Task evaluates semantic memory over a LIFO ordered structure. During the memory formation phase, a sequence of `push` and `pop` operations is delivered to the agent one at a time as observations of the form "`Step 10/20: Execute push(7)`". `push(x)` adds a value to the top of the stack and `pop()` removes the top element. A maximum size cap prevents unbounded growth, and `pop()` is only attempted when the stack is non-empty. The agent must simulate the stack state across the full sequence. During the probe phase, the agent is asked to return the value at a specific index position using the final stack state stored in its memory.

Queue Simulation Task evaluates semantic memory over a FIFO structure and is structurally similar to the stack task. During the memory formation phase, `enqueue` and `dequeue` operations are delivered in the format "`Step 10/20: Execute enqueue(3)`". The `enqueue(x)` operation appends a value to the back of the queue and `dequeue()` removes the front element. During the probe phase, the agent is queried for the value at a specific index from the final queue and must respond from memory alone. Together, the stack and queue tasks isolate sequential state tracking.

Graph Reasoning Task evaluates memory over a dynamic relational structure. During the memory formation phase, an undirected graph over N nodes is initialized with randomly

sampled integer node values and an empty edge set. Updates are delivered sequentially as "`Update 10/20: Add edge(N02, N12)`" or "`Update 11/20: Set Value(N07) = 10`". Three types of operations are performed: `set_value` overwrites a node's value, `add_edge` introduces an undirected edge between two nodes, and `remove_edge` removes an existing edge. The agent must maintain all node values and the adjacency structure within its memory.

During probe time, no graph information is provided and the agent must rely entirely on its internal representation of the graph. One probe mode is active per experiment across four difficulty levels:

- **Connected Direct (Edge Presence, Episodic):** Given two nodes, the agent must return 1 if a direct edge exists between them and 0 otherwise.
- **1-Hop Filtered Sum (Semantic):** Given a center node and a numeric threshold, the agent must return the sum of values of all neighboring nodes whose value meets or exceeds the threshold.
- **Shortest Path Distance (Semantic):** Given a source and destination node, the agent must return the minimum hop distance between them, or -1 if no path exists.

I.2 Task Type Classification

Episodic Tasks: Name-Attributes, Graph Edge Presence, and Inventory Ledger (Snapshot Mode). These tasks satisfy definition 4 because they rely on the retention and retrieval of independent, discrete facts. In the Name-Attributes task, facts such as "Alice is 34" and "Bob lives in Toronto" do not influence one another, resulting in near-zero multi-information where $I(A_1, \ldots, A_n) \leq \epsilon$. The probe questions demand the recall of these isolated facts, overwhelmingly favoring lookup queries such that $P_{q\sim P_Q}(q \in \mathcal{Q}_{\text{lookup}}) \geq 1 - \delta$. Similarly, the Edge Presence (Connected Direct) task asks whether a specific edge exists, which can be resolved by looking up the most recent discrete `add_edge`/`remove_edge` observation for that pair without structural traversal. When the Inventory Ledger operates in snapshot mode, queries can be answered by retrieving the most recent authoritative state observation for an item ID, again functioning as an independent key lookup.

Semantic Tasks: Stack, Queue, Graph 1-Hop Neighborhood Sum, Graph Shortest Path Distance, and Inventory Ledger (Delta Mode). These tasks satisfy definition 5 because the correct answer cannot be found in any single isolated observation and must be computed from the relationships or aggregation of observations. In the Stack and Queue tasks, operations like `push`, `pop`, `enqueue`, and `dequeue` are highly interdependent: the final state is a compressed joint representation of the full sequence, meaning the joint entropy of the active state is strictly bounded by a small fraction of the sum of step entropies, $H(A_1, \ldots, A_n) \leq \epsilon \sum_{i=1}^{n} H(A_i)$. Probing an index requires evaluating the operational history jointly. Likewise, the Graph 1-Hop Neighborhood Sum and Shortest Path Distance tasks require traversing adjacency structures and aggregating values across multiple nodes, guaranteeing that queries are relational ($P_{q\sim P_Q}(q \in \mathcal{Q}_{\text{relational}}) \geq 1 - \delta$). Finally, the Inventory Ledger in delta mode acts as a semantic task because current inventory quantities cannot be looked up verbatim and require integrating an interdependent stream of stock additions and removals into a joint state encoding.

I.3 Agents

We evaluate 6 agent architectures that differ in how they form, store, and retrieve memories.

Everything-in-Context (EiC) represents the uncompressed semantic agent baseline. It maintains a single growing string of chat history that is prepended to every new prompt in full. During the memory formation phase, each observation and its corresponding response are appended verbatim to the memory buffer. At the probe phase, the entire memory buffer is included in the prompt and the agent generates a response conditioned on the entire context. Memory formation is lossless without compression or selection, and effective capacity is bounded by the context window and available VRAM.

Specialized-in-Context maintains an explicit, editable memory buffer that is rewritten by the LLM after each turn. Upon receiving an observation, the agent first acknowledges the step conditioned on the current memory, then issues a separate memory update call in which the model is prompted to produce a concise, structured rewrite of the memory buffer incorporating the new observation. The model is provided with a task-specific structural hint (e.g., maintaining an explicit key-value list for episodic tasks vs. rewriting the active array, ledger, or adjacency table in-place for semantic tasks). To guard against catastrophic memory collapse, the agent applies a length-ratio guard that rejects updates exhibiting excessive shrinkage in character count and falls back to the prior memory state when triggered. At probe time, the frozen memory string is placed into the prompt.

Retrieval-Augmented Generation (RAG) maintains a persistent vector database using the Qwen3-0.6B embedding model. During the memory formation stage, the agent issues a storage decision call in which the model extracts and summarizes the factual content of the observation for storage. The resulting entry is converted to a vector embedding and written to the database. At both memory formation and probe phases, the agent retrieves the top-k semantically similar entries from the database and injects them into the prompt as retrieved context. Storage is skipped when the model returns an empty string or `None`.

HippoRAG implements graph-based memory following Gutiérrez et al. (2024). During formation, a dedicated extraction prompt maps each observation to subject–relation–object triples that update a NetworkX knowledge graph; numeric node values (e.g., `Set Value(N)=V`) are stored as edges to literal value nodes, making them full participants in the graph topology. At probe time, seed nodes are identified via embedding similarity and Personalized PageRank (PPR) retrieves the highest-scoring subgraph as context.

HippoRAG with Attributes is a semantic variant of HippoRAG that stores numeric node values directly as node attributes on entity nodes rather than as edges to literal value nodes. Keeping numeric literals off the graph topology prevents high-degree literal value nodes from distorting Personalized PageRank propagation and binds node values jointly with structural entity connectivity.

Attention Sink keeps the same growing chat transcript as Everything-in-Context, but performs generation through a HuggingFace `sink_cache` path (Xiao et al., 2023) with preserved initial sink tokens and a sliding window, mitigating KV-cache memory growth during generation on long episodes.

I.4 Analysis of Popular Memory Mechanisms

RAG is episodic. In the cell-probe model, computation is free and the cost of an operation is the number of memory cells probed. RAG stores each discrete chunk of data in an isolated cell per turn ($I(A_1, \ldots, A_n) \approx 0$ at the storage layer). A retrieval query embeds the input as $q = \phi(\text{query})$ and selects the top-k chunks via maximum inner product search (MIPS), where k is a fixed system constant ($k = 5$). Because the inner-product scoring is internal computation, the cell-probe read cost consists of fetching the k retrieved chunks, costing $O(k) = O(1)$ cell probes. Inserting or deleting a chunk modifies $O(1)$ cells while total space scales as $\Theta(\sum_{i=1}^{n} H(A_i))$. Therefore, RAG satisfies definition 6.

Everything-in-Context is semantic. The standard in-context agent utilizes self-attention over its context window to dynamically construct joint representations of the memory state. By computing pairwise attention scores across all tokens, the mechanism maintains a contextualized joint encoding of the stored variables $A_1, \ldots, A_n$ across attention layers, supporting relational queries $\mathcal{Q}_{\text{relational}}(\Phi)$ directly within the forward pass without isolated per-record scanning.

Specialized-in-Context adapts across semantic and episodic regimes. Under episodic task hints, the agent appends observations as discrete key-value entries in the buffer ($I(A_1, \ldots, A_n) \approx 0$). Under semantic task hints, the agent rewrites and compresses the state

into a compact joint encoding (e.g., current stack/queue state or active inventory totals), achieving $O(H(A_1, \ldots, A_n))$ space (definition 7) at the cost of reading and rewriting $\Omega(k)$ shared state tokens on every update (theorem 2).

HippoRAG is episodic. HippoRAG extracts discrete subject–relation–object triples $\{t_1, \ldots, t_n\}$ from independent observations in isolation, yielding $I(A_1, \ldots, A_n) \approx 0$ at storage time and total space $H(G) \approx \sum_{i=1}^{n} H(t_i)$. Numeric values are appended as edges to literal value nodes rather than updating a compressed state variable, and probe-time retrieval relies on localized top-k seed matching followed by fixed-budget subgraph extraction.

HippoRAG with Attributes is semantic. By binding numeric values directly as mutable node attributes on structural entities and keeping literals off the graph topology, HippoRAG with Attributes maintains a joint encoding of entity state and connectivity ($H(A_1, \ldots, A_n) < \sum_{i=1}^{n} H(A_i)$) while preserving undistorted Personalized PageRank traversal over the relational graph structure.

Attention Sink is semantic. Attention Sink applies self-attention over a preserved set of initial sink tokens and a sliding window of recent tokens, maintaining a joint contextual representation within the active attention window while bounding generation KV-cache size.

I.5 Agent Parameters

All agents use **Qwen3-4B** as the generative backbone (`qwen3:4b` via Ollama; `Qwen/Qwen3-4B` via HuggingFace, `bfloat16`, `device_map=auto`). All runs use temperature 0.2 and `max_new_tokens=2048`. Thinking-mode `<think>` blocks are stripped before scoring or memory updates. Agent-specific hyperparameters are listed below:

- **Specialized-in-Context:** Memory guard rejects rewrites when character-length ratio falls below ≥ 0.6 (guard activates once prior memory exceeds 200 characters). Task-specific structural prompts instruct the model to maintain an explicit entity–attribute table (Name-Attributes), running item quantity/status ledger (Inventory Ledger), ordered array of active elements (Stack/Queue), or current node-value and adjacency list (Graph tasks).
- **RAG:** Retrieval top-$k = 5$; embeddings use `Qwen3-0.6B` (Ollama) or `all-MiniLM-L6-v2` (HuggingFace); fresh ChromaDB collection initialized per experiment run.
- **HippoRAG and HippoRAG with Attributes:** Both variants use identical retrieval and graph hyperparameters: `seed_k` = 3, `ppr_top_n` = 15, PageRank `max_iter` = 200, and the same embedding models as RAG; the NetworkX graph and entity index are reset per experiment.
- **Attention Sink:** `window_length` = 16,384 and `num_sink_tokens` = 4, implemented via `transformers-community/sink_cache` (`transformers` 4.52–4.53.x).

I.6 Task Parameters

Table 2 lists the scaling parameters used at each difficulty level L1–L4. Results are averaged over ten random seeds (`20001`–`20005` and `30001`–`30005`). Graph experiments use `edge_add_prob` = 0.35 throughout; `edge_remove_prob` varies by level for the connected-direct probe mode (see Table 2) and is fixed at 0.15 for filtered 1-hop sum and shortest-path modes. Inventory Ledger uses the same numeric scale for both delta and snapshot observation modes.

Reproducing results. Full setup instructions and run scripts are provided in the repository README at `https://anonymous.4open.science/r/MemoryExp-8B6F/README.md`.

Table 2: Task parameters per difficulty level. Stack/queue probe count equals final structure size (`num_probes` = −1). †`edge_remove_prob` varies per level for connected-direct: 0.20 / 0.22 / 0.25 / 0.20.

Task	Parameter	L1	L2	L3	L4
Stack / Queue	`num_ops`	5	20	50	100
	`max_size`	20	20	20	20
Name-attributes	`num_entities`	20	30	40	150
	`num_facts`	80	120	200	600
	`num_probes`	80	100	100	250
Graph (filtered sum / shortest path)	`num_nodes`	8	14	20	30
	`num_updates`	20	45	70	100
	`num_probes`	20	35	50	80
Graph (connected direct†)	`num_nodes`	14	20	30	60
	`num_updates`	50	100	150	300
	`num_probes`	50	80	100	200
Inventory ledger (delta / snapshot)	`num_items`	5	8	12	20
	`num_updates`	10	20	40	80
	`num_probes`	10	20	40	80

I.7 Evaluation Metrics

Each task uses a deterministic scoring function that maps the agent output to a scalar value in $[0, 1]$, representing the proportion of probes answered correctly during the frozen-memory probe phase.

For the **Name-Attributes** task, the scoring function computes the exact match between the agent's response and the ground-truth value. For numeric fields (`age`, `height_cm`), the scorer searches for the first integer token anywhere in the response to accommodate conversational phrasing. For categorical and string fields (`location`, `occupation`, `hobby`, etc.), the scorer applies text normalization (lowercasing, replacing non-alphanumeric characters with spaces, and stripping extra whitespace) and marks the response correct if the normalized output matches the ground truth exactly.

For the **Inventory Ledger** task, current-quantity probes require a single integer token matching the ground truth exactly, while current-status probes require a single categorical token (`active`, `hold`, or `discontinued`) matched case-insensitively via regular expressions.

For the **Stack** and **Queue** tasks, if the final structure is empty, the response is marked correct if and only if the token `EMPTY` appears (case-insensitively). For non-empty states, each probe specifies an index from the bottom of the stack or front of the queue, and the scorer extracts the first integer found in the response to compare against the ground truth.

For the **Graph Reasoning** task, strict single-integer parsing is applied across all three probe modes: the stripped response must match the ground-truth integer (0/1 for Connected Direct, arithmetic sum for 1-Hop Filtered Sum, and minimum hop count or −1 for Shortest Path Distance).

J Compute Resources and Memory Footprint Scaling

All experiments ran on a single NVIDIA H100 80GB MIG slice (`3g.40gb`) per job on a university cluster. Wall-clock time per experiment ranged from approximately 0.5 to 1.7 hours depending on task and difficulty level, totaling approximately 300 GPU-hours.

fig. 4 reports the empirical peak VRAM overhead (peak − post_agent, in MiB on a log scale) across difficulty scales L1–L4 for all eight diagnostic tasks, validating the space–time complexity trade-offs established in theorem 2 and lemma 6. Note that *HippoRAG with Attributes* exhibits the exact same flat $O(1)$ active VRAM profile ($\sim$800–1,200 MiB) as base

HippoRAG (`hipporag`), as storing node values as graph attributes adds negligible memory overhead relative to the embedding index and LLM backbone.

K Meta-Analysis Details

K.1 Result Analysis

As detailed in table 1, four of the five benchmarks reach statistical significance ($\alpha = 0.05$). LongMemEval_S (Wu et al., 2024) provides the strongest evidence ($p < 10^{-4}$), with memory mechanism–task alignment accounting for nearly 69% of score variance ($\eta^2 = 0.6869$). LoCoMo (Maharana et al., 2024), Episodic Memory Eval (Huet et al., 2025), and MemBench (Tan et al., 2025) all yield statistically significant effects ($\eta^2 \approx 0.086$–0.115). The sole null result is Memory AgentBench (Hu et al., 2025) ($p = 0.8563, \eta^2 = 0.0006$); however, its headline evaluations aggregate conflicting episodic and semantic sub-tasks concurrently, masking regime-specific advantages. Overall, these findings confirm that our episodic/semantic taxonomy explains substantial variance in existing benchmark performance.

K.2 Meta-Analysis Setup

Candidate benchmarks were identified through searches on Semantic Scholar and arXiv covering publications from 2023 through 2026. From the resulting publications, benchmarks were selected based on the inclusion criteria in appendix K.3. The final set comprises five benchmarks from peer-reviewed venues:

- **MemBench** (Tan et al., 2025) uses a two-level framework to distinguish factual memory from reflective memory across participation and reflection interaction modes, measured via participation accuracy.
- **LoCoMo** (Maharana et al., 2024) generates long multi-session dialogues grounded in persona event graphs and evaluates single-hop QA, multi-hop QA, event summarization, and temporal reasoning using F1 score.
- **LongMemEval** (Wu et al., 2024) evaluates agents across up to 500 sessions spanning five memory abilities: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention, scored using GPT-4o as a judge.
- **Episodic Memory Eval** (Huet et al., 2025) generates events grounded in specific dates, locations, entities, and contexts across over 650 question pairs, measured using accuracy and F1 score.
- **Memory AgentBench** (Hu et al., 2025) evaluates agents across accurate retrieval, test-time learning, long-range understanding, and conflict resolution as knowledge accumulates over time, reporting accuracy.

Score Normalization and Regime Assignment. Because raw evaluation metrics vary across benchmarks and sub-tasks (e.g., F1 vs. accuracy), raw scores within each benchmark sub-task are standardized (min-max normalized to $[0, 100]$ within each task group) prior to running the one-way ANOVA so that scale differences across metrics do not confound the alignment test. Each benchmark sub-task is classified as *Episodic* (definition 4) if queries test isolated fact retrieval (e.g., single-hop QA, direct factual recall) and *Semantic* (definition 5) if queries require multi-session synthesis, temporal state tracking, summarization, or conflict resolution. Similarly, evaluated baseline methods are classified as *Episodic* (definition 6) if they store and retrieve isolated chunks or triples via top-k lookup (e.g., vector RAG, graph triple lookup) and *Semantic* (definition 7) if they maintain a joint context representation or iterative state compression. A sample of the dataset is provided in Table 3, and the complete dataset is available at `https://anonymous.4open.science/r/MemoryExp-8B6F/utils/Memory%20Types%20Meta%20Analysis.csv`.

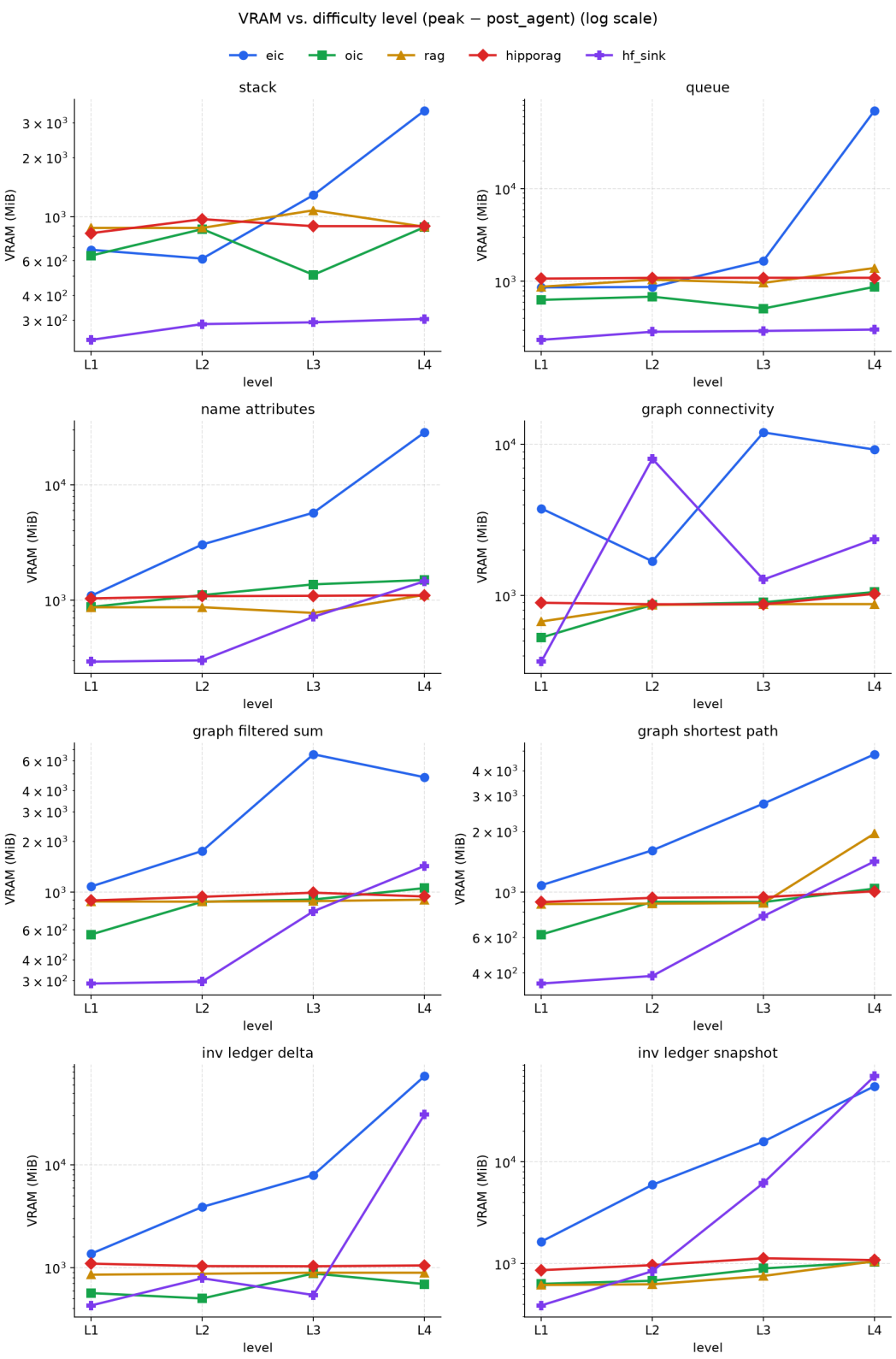


Figure 4: Peak VRAM overhead (peak − post_agent, in MiB on a log scale) across difficulty levels L1–L4 for all eight diagnostic tasks. Legend mapping to paper terminology: `eic` = Everything-in-Context; `oic` = Specialized-in-Context; `rag` = RAG; `hipporag` = HippoRAG (and HippoRAG with Attributes); `hf_sink` = Attention Sink. Episodic stores (`rag`, `hipporag`) and state-rewriting compression (`oic`) maintain flat $O(1)$ active VRAM footprints ($\sim$800–1,200 MiB), whereas uncompressed context (`eic`) grows super-linearly to $>$ 30 GiB at L4, causing the out-of-memory budget violations analyzed in section 6.

Table 3: First ten rows of the meta-analysis dataset. `Classification` = algorithm memory type classification (definitions 6 and 7). `Task Mem Type` = memory regime required by the benchmark sub-task (definitions 4 and 5).

Algorithm	Algo Mem Type	Classification	Benchmark	Task	Task Mem Type	Metric	Score
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, multi hop	Semantic	F1	19.400000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, single hop	Episodic	F1	46.300000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, multi hop	Semantic	F1	26.800000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, single hop	Episodic	F1	48.100000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, multi hop	Semantic	F1	36.100000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, single hop	Episodic	F1	44.300000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, multi hop	Semantic	F1	30.600000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, single hop	Episodic	F1	42.200000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, multi hop	Semantic	F1	30.500000
GPT 3.5 Turbo 16k	RAG	Episodic	LoCoMo	Q and A, single hop	Episodic	F1	44.600000

K.3 Inclusion Criteria

Benchmarks were selected based on three criteria: (1) the evaluation code must be publicly available, enabling independent verification of task design and evaluation; (2) the dataset must be publicly released to analyze and assign episodic or semantic labels to individual sub-tasks; and (3) the benchmark must evaluate memory architectures and tasks that map onto the episodic/semantic taxonomy. Of the eight benchmarks satisfying our initial search, three were excluded from formal ANOVA testing: RealTalk ($N_{\text{mis}} = 0$), and RealMem and HaluMem, which entangle multiple memory regimes within individual queries in an ambiguous way that prevents clean classification. All 3 excluded benchmarks contained ≤ 6 entries each and were therefore susceptible to small-sample noise.

K.4 Limitations

Several limitations of this meta-analysis set should be noted. The five benchmarks differ in underlying task domains and raw evaluation metrics (F1 in LoCoMo and Episodic Memory Eval, participation accuracy in MemBench, GPT-4o judge score in LongMemEval_S, and accuracy in Memory AgentBench). Consequently, we evaluate one-way ANOVA alignment effects separately within each benchmark (Table 1) and report the median variance explained (η^2) across benchmarks rather than pooling unnormalized raw scores across disparate metrics.