Experiential Hierarchical Memory for Long-Horizon Intelligence

Chris Riley, AxR Lab

Abstract

Long-horizon intelligence requires systems to retain, generalize, and revise information over time. Humans accomplish this through hierarchical and abstract memory structures, forgetting and reconstructing details as needed to allow for prolonged experience gain without unbounded storage. Existing approaches to long-term memory in AI systems, such as context prefill (including RAG) and parametric updates, do not accomplish this. Consequently, in a recent assessment of progress towards Artificial General Intelligence, "long-term memory storage" was assessed at 0% capability; this was the only category for which GPT-5 received a 0% score (Hendrycks, 2025).

We propose Experiential Hierarchical Memory (EHM), a vector-first external memory architecture that separates episodic recall, multi-episode abstraction, hierarchical selection, and provenance-based disambiguation while treating language as a derived rendering layer rather than as the storage substrate, preserving maximum information in memory while also leaving open a future possibility of extending the architecture into multimodal memory. Memory retrieval begins with abstract nodes and progresses to more granular memories where they are useful in the context at hand. Consolidation and pruning algorithms form abstractions dynamically and reduce unused granular memory data, enabling both preservation of useful memory and performance improvements, particularly in dense and adversarial contexts.

In controlled synthetic experiments scored against hidden world states, we show that hierarchical memory with abstraction-first retrieval substantially outperforms flat mixed (abstract and episodic memory) retrieval on structured downstream revision outputs, with the gap widest on adversarial cases requiring conflict resolution and provenance-sensitive evidence recovery. This indicates that hierarchical memory is usable and in fact beneficial, particularly when facing high distraction. Across a 10-seed matrix, hierarchical retrieval reaches 0.9167 ± 0.0833 adversarial exact success versus 0.2000 ± 0.0928 for flat mixed retrieval; the per-seed ranges

do not overlap, and single-memory baselines achieve zero. Under dense memory overload, staged episodic pruning not only preserves this advantage but can improve over the unpruned baseline — the pruned system exceeds the full dense store by up to +0.18 in mean success across a broad retention plateau from 50% down to 10% of the episodic budget — because hierarchy-aware pruning acts as interference reduction, not just compression. The hierarchy advantage also survives into a tool-mediated LLM-in-the-loop setting, where a frozen Llama-3.2-3B-Instruct model using explicit cognitive memory tools reaches 0.9500 adversarial exact success under the arbitrary-depth hierarchy compared to 0.7500 for flat mixed, confirming that the gain is not an artifact of deterministic output construction.

These results use tool mediation for LLM adaptation and explicit memory access. Initial tests of implicit conditioning, including prefix fills, LoRA readers, and cross-attention readers, all failed to achieve equivalent performance. Direct model conditioning, de-consolidation, and scaled evaluation are the immediate next steps.

1. Introduction

Language models increasingly rely on external memory to support long-horizon reasoning. Existing memory integrations for language models usually fall into two broad families. In parametric approaches, new information is absorbed into weights, adapters, LoRA modules, or other parameter-space mechanisms (De Cao et al., 2021; Mitchell et al., 2022; Meng et al., 2023). In context-based approaches, retrieved information is appended to the prompt as largely undifferentiated text (Lewis et al., 2020; Borgeaud et al., 2022). Recent work on tree-, graph-, and long-term-memory retrieval has emerged in part because conventional flat retrieval struggles on holistic, global, or multi-hop questions that require organization across experiences rather than recovery of one local chunk (Sarthi et al., 2024; Edge et al., 2024; Gutiérrez et al., 2024). But the state of the art is insufficient. Long-term memory storage remains an unsolved problem, and all current approaches ultimately lose fidelity and performance through parameter dilution, context collapse, or the increasing difficulty and inefficiency of navigating unbounded storage requirements.

We propose Experiential Hierarchical Memory (EHM), a different approach. Memories remain explicit external objects but are not treated as interchangeable context chunks. Instead, the system distinguishes memories at variable layers of abstraction, building up abstract memories as consolidations of episodic memories and/or other abstract memories; preserves links between memories; and retrieves relevant and useful memories for inference conditioning at variable abstraction layers through hierarchy-aware selection and composition. First, root abstract memories are selected for relevance to the context at hand, then child memories in the

memory graph are prioritized for further consideration. Pruning allows for memory conservation without significant information loss, and in some cases producing output improvements as distracting details are trimmed. In the current implementation, memory nodes are vector-first latent objects; text is rendered only when a human-readable or frozen-model-facing view is needed.

Moment-in-time testing is inherently limited for evaluating an architecture whose principal purpose is long-horizon utility. Nevertheless, we can test core design assumptions. We seek to determine whether explicit hierarchical memory structure yields gains that cannot be reduced to either parametric storage or flat retrieval-into-context, especially on tasks that require combining defaults, overrides, provenance, and supporting evidence. This goes beyond the claim that hierarchical memory will retain more information over time, and seeks to demonstrate that hierarchy, consolidation, and pruning improve outcomes even in the near term. In other words: does the organization of memory matter, or just the content?

Specifically, we test a systems hypothesis: hierarchical memory should enable behavior that flat memory does not, visible not only in retrieval diagnostics but also in structured downstream revision outputs. Revision tasks require default traits to be overridden in specific circumstances, combining episode-specific data with general tendencies, making them well suited for this evaluation. We test across three settings: deterministic structured-output construction, tool-mediated LLM generation, and memory-budget stress tests under dense overload.

This architecture models itself after human cognition in several respects. Pruning, for instance, reduces interference and reflects the brain's forgetting functions that enable long-term durable memory. The broader motivation for this work draws on a conceptual view of abstraction and resilience as central to intelligence and consciousness (Riley, 2026). The present paper is narrowly focused on a potential architecture for abstract memory evaluated through controlled empirical tests, intended as a starting point for richer memory implementation and testing at larger scales and on diverse tasks.

2. Method

The principal goal of this paper is to test the viability and utility of hierarchical memory. We evaluate this through structured generation: using memory to construct artifacts that match hidden world states in response to natural language queries. This tests whether hierarchical memory architecture helps or hinders compared to other approaches to organizing memory, under conditions of increasing complexity and interference. We first test with deterministic constructors to isolate the memory architecture's contribution, then with frozen Llama models accessing memory through tool calls to confirm the result survives neural generation.

The EHM pipeline is vector-first: symbolic state is encoded into latent vectors, stored and consolidated in vector space, and rendered into text only at the final compatibility boundary. The system operates in six stages: world-state generation, surface rendering, vector encoding, latent storage and consolidation, routing and retrieval, and downstream output construction.

2.1 Hidden World State

Each benchmark instance begins from a symbolic `WorldState` containing three latent object types:

- **Traits**: stable default preferences for a task type (e.g., preferred tone, structure, or length).
- **Facts**: concrete episodic information bound to a specific entity and source (e.g., a deadline, a repository branch, a check-in date).
- **Overrides**: entity-specific modifications to a default trait.

These latent objects are never scored through surface strings. Evaluation is performed against the hidden world state, not against labels produced by the renderers. This anti-collusion design ensures that high scores reflect recovery of latent structure rather than memorization of generator phrasing.

2.2 Surface Rendering and Vector Encoding

The benchmark includes two channel-separated surface renderers — one for episodes, one for queries — useful for integrity audits and text-based interfaces. However, the operational path bypasses surface text: each episode's symbolic payload is encoded into a deterministic fixed-dimensional vector, and queries are likewise vector-encoded. Surface channels sit beside the memory path, not inside it.

Two realization families (`a` and `b`) are implemented. Under the vector-first pipeline they serve as an invariance check: identical metrics across families confirm the system is not relying on surface phrasing.

2.3 Memory Representation

The memory store contains two node types. **Latent episodic memories** carry an L2-normalized vector, symbolic slot payload, task type, entity list, and structural metadata. **Latent abstract memories** carry a centroid vector plus links to constituent child episodes. Abstractions do not carry any of the specific details of individual episodes; they reflect patterns among them. The vector is the operational representation; symbol slots are retained for rendering and inspection.

Abstraction, in principle, can be conducted on a range of digital substrates. Some LLMs use compressed context representations as a means of streamlining. Transformer heads in large language models tend to encode more abstract representations at higher layers and more concrete representations at lower layers. Yet no current approaches create explicit hierarchical relationships between memory objects, which is the novel contribution of this work. We use vectors as a substrate to facilitate this, as vectors allow the system to easily compute and manage hierarchies and thus test our central hypothesis that hierarchy itself adds value.

Abstract memories in the current architecture are centroid vectors — summary statistics over their child episodes. They preserve navigational information (where in memory space to look) but not referential content (what specific evidence was there). This means pruning is effectively irreversible: the vector difference between pre- and post-pruning centroids reflects what was lost in aggregate but cannot reconstruct the discrete symbolic content of pruned episodes. Richer abstract representations — learned projections rather than summary statistics, capable of supporting reconstruction as well as navigation — are a natural next step. The current centroid-based design is sufficient to demonstrate the value of hierarchy and pruning-as-interference-reduction, but it does not yet capture the generative, schema-like quality of human abstract memory. Nevertheless, even with this lossy form of abstraction, we show strong performance in our designated tasks; in other words, in some or perhaps many environments, remembering or reconstructing the details is simply not necessary.

2.4 Consolidation

Consolidation converts latent episodes into abstract memories: episodes are grouped by task type, pairwise cosine similarity is computed, connected components are formed under a fixed threshold, and sufficiently large clusters become centroid abstract memories. Because of the nature of the latent vectors in episodes, the similarities are substantive in nature, albeit opaque; the abstract memory thus reflects a meaningful pattern in underlying episodes. The consolidator operates on latent vectors and structural metadata only — it does not read evaluator-side tags or surface strings. The mechanism is intentionally simple; we show that even lightweight vector-space abstraction supports strong hierarchical retrieval gains.

Initial testing used a two-tier approach with a single layer of abstraction. Subsequent testing expanded this to arbitrary depth, extending consolidation to include abstract-over-abstract consolidation and preserving a root tier of nodes without parent relationships. Retrieval thus begins with root abstract nodes and descends to children as relevance thresholds are met.

2.5 Routing

A lightweight rule-based router selects a retrieval strategy from { `episodes_only` ,

`abstract_only`, `mixed`} based on symbolic query slots: fact queries route to episodes, trait queries to abstractions, revision queries to mixed. No route hints are injected. The flat baseline `mixed_flat` is used only as a forced ablation at evaluation time, designed to isolate the specific gains from hierarchical organization as distinct from the gains of simply having access to both memory types (which the comparisons to `episodes_only` and `abstract_only` address). The rule-based router is sufficient for the current synthetic benchmarks; gains from hierarchy are achieved despite, not because of, routing sophistication.

2.6 Retrieval

We compare four retrieval strategies, all scored by cosine similarity in latent space:

`episodes_only`. Top-ranked episodic memories, with a global score floor to suppress weak same-task distractors.

`abstract_only`. Top-ranked abstract memories.

`mixed_flat`. Top-ranked episodes and abstractions retrieved independently. Both memory types are available, but hierarchical relationships are not exploited.

`mixed` (hierarchical). The main architectural contribution. Retrieval proceeds in four steps: (1) select the top abstract memories; (2) reserve global episode slots for revision-critical evidence (current facts, overrides); (3) descend into child episodes of selected abstractions, preferring representative children that provide supporting evidence; (4) fill remaining slots from the global episode ranking.

The key difference from `mixed_flat` is step 3: abstraction-first child descent converts retrieval from a flat nearest-neighbor problem into a structured selection problem. The system must recover not only the right default and override, but also the supporting evidence that justifies the default — evidence that may not rank highly on query similarity alone.

2.7 Downstream Structured Output

For revision cases, the system constructs a deterministic structured artifact from retrieved memories containing nine fields: entity, current fact kind and value, evidence source, default trait, default evidence, applied override, final decision, and recommendation. The artifact is initially built by deterministic rules rather than a neural generator, isolating the memory architecture's contribution from generation quality confounds. Subsequently, we introduce a neural generator to show that the gains of hierarchy extend to that setting. Scoring reflects retrieval completeness rather than generation fluency.

3. Experimental Setup

3.1 Benchmark Generation

All results are produced from synthetic benchmarks built on hidden latent world states. Each run samples a `WorldState` containing default traits, entity-bound facts, and entity-bound overrides, rendered into episodes and queries through the channel-separated procedure described in Section 2.2. The generated suite contains four case families: **episodic** (concrete fact retrieval), **abstract** (task-level default retrieval), **revision** (integrating a fact, default trait, and override), and **adversarial revision** (revision augmented with distractor facts and overrides to create interference). Adversarial revision is the hardest test and produces the largest separation between strategies.

3.2 Anti-Collusion Design

Three safeguards prevent surface-level shortcuts: (1) all metrics are computed against hidden world-state truth, not surface labels; (2) no route hints are injected; (3) two independently phrased realization families (`a` , `b`) serve as an invariance check — identical results across families confirm the system is not relying on surface form.

3.3 Compared Conditions

We compare four retrieval strategies addressing two questions:

Question	Comparison
Are episodic and abstract memories functionally distinct?	`episodes_only` vs. `abstract_only`
Does hierarchy help beyond having both memory types?	`mixed_flat` vs. `mixed`

The first question validates that the memory types serve distinct functional roles. The second is the central test: `mixed_flat` is the critical control because it has access to the same memory content as hierarchical `mixed` but does not perform abstraction-first child descent. Any gap between these two is attributable to hierarchical structure, not memory budget.

3.4 Metrics

Latent evaluation checks whether retrieval recovered the relevant hidden symbols (facts, traits, overrides, provenance). Episodic questions are answered only from episodes; trait questions only from abstractions, enforcing functional separation.

Downstream evaluation scores each artifact field-by-field against hidden truth. The primary metric is **adversarial revision exact success** — the fraction of adversarial cases where all fields are correct — because this is where the hierarchy gap is most pronounced.

3.5 Evaluation Protocol

The main downstream results use a 10-seed matrix (seeds: 11, 17, 23, 29, 31, 37, 41, 43, 47, 53), each producing $n_{revision} = 12$ and $n_{adversarial} = 12$ cases. All four strategies are evaluated on the same cases per seed. Under the vector-first pipeline, aggregate metrics are identical across realization families. Reported values are means across seeds.

4. Results

4.1 Capability Decomposition

Episodic and abstract memory play distinct functional roles. On episodic cases, `episodes_only` succeeds while `abstract_only` fails; on abstract cases, the reverse. On revision cases, which require understanding both traits and facts, neither single-memory condition is sufficient. Routing accuracy is 1.0 and results are identical across realization families, confirming the benchmark measures latent structure recovery.

Table 1, Panel A: Capability Decomposition (Latent Recovery).

Strategy	Episodic	Abstract	Revision	Adversarial Revision
`episodes_only`	YES	NO	NO	NO
`abstract_only`	NO	YES	NO	NO
`mixed_flat`	YES	YES	YES	PARTIAL
`mixed`	YES	YES	YES	YES

The two memory types are not weaker versions of each other — they contribute structurally different information. This merits explicit confirmation: mixed retrieval is the first condition with access to both ingredients needed for revision.

4.2 Hierarchy Versus Flat Mixed Retrieval

Having both memory types is necessary but not sufficient. On `adversarial_revision`, `mixed_flat` reaches only 0.5000 exact case success while hierarchical `mixed` reaches 1.0000; single-memory baselines remain at 0.0000. The gain comes from abstraction-first child descent: the retriever selects an abstract node, then descends into supporting child episodes while preserving globally relevant fact and override evidence. This converts retrieval from a flat nearest-neighbor problem into a structured selection problem, and the improvement is evidence for the causal role of hierarchy, not merely of having both memory types.

4.3 Downstream Structured Revision Outputs

The strongest bridge from memory architecture to task outcome is the downstream revision benchmark, where the system emits a final artifact scored field-by-field against hidden truth. Across the 10-seed matrix, hierarchical `mixed` substantially outperforms all alternatives (Table 1, Panel B). The most informative slice is adversarial revision: `mixed_flat` reaches 0.2000 ± 0.0928, while `mixed` reaches 0.9167 ± 0.0833. Single-memory baselines achieve zero.

Table 1, Panel B: Downstream Structured Revision (Exact Success). Mean ± SD over 10 seeds ($n*text{revision}=12$, $n*\text{adversarial}=12$ per seed). Identical across realization families `a` and `b`.

Strategy	Overall	Adv. Revision
`episodes_only`	0.00 ± 0.00	0.00 ± 0.00
`abstract_only`	0.00 ± 0.00	0.00 ± 0.00
`mixed_flat`	0.56 ± 0.06	0.20 ± 0.09
`mixed`	**0.96 ± 0.04**	**0.92 ± 0.08**

The gap is stable across seeds: the best `mixed_flat` adversarial seed is 0.3333, the worst `mixed` seed is 0.75 — the distributions do not overlap. `mixed_flat` typically recovers the right fact, default, and override but loses the support structure needed for full correctness; hierarchical descent closes that gap.

4.4 Tool-Mediated LLM Revision Outputs

Does the hierarchy advantage survive when a neural language model produces the final output? Prior attempts via rendered text briefs, latent prefix conditioning, and adapter-based readers all failed to preserve the hierarchy signal. We report a positive result using explicit tool-mediated memory access.

A frozen `Llama-3.2-3B-Instruct` model interacts with EHM through four cognitive tools — `recall_relevant_experiences`, `summarize_what_is_typical`, `check_for_exceptions_or_changes`, and `trace_where_that_came_from` — calling them to inspect JSON memory objects and emit the same scored revision artifact. No hidden memory conditioning is applied.

Table 2: Tool-Mediated LLM Revision (Exact Success). Top: pooled across initial 8-seed evaluation. Bottom: 5-seed confirmation after arbitrary-depth hierarchy refactor.

Evaluation	Strategy	Overall	Revision	Adv. Revision
Pooled initial (8 seeds)	`mixed_flat`	0.88	1.00	0.75
	`mixed`	**0.89**	0.91	**0.88**
Post-refactor (5 seeds)	`mixed_flat`	0.88	1.00	0.75
	`mixed`	**0.95**	0.95	**0.95**

The hierarchy advantage is preserved and strengthened after the arbitrary-depth refactor. On adversarial cases — requiring conflict resolution, distractor rejection, and provenance-sensitive integration — hierarchical `mixed` consistently outperforms `mixed_flat`. On ordinary revision cases, `mixed_flat` is at ceiling while `mixed` is slightly lower, suggesting the LLM layer adds synthesis noise on easier cases. The positive result comes from explicit tool mediation, not from hidden prompt absorption or latent conditioning.

4.5 Pruning Stress Tests

A system that accumulates experience must manage memory budgets. We test whether EHM remains effective under staged episodic pruning, where low-access episodes are progressively removed and abstract nodes are repaired after child deletion.

Table 3: Pruning Stress Tests (Hierarchical Mixed Retrieval). Stable pre-cliff plateau; delta relative to unpruned dense baseline.

Workload	Unpruned	Plateau Range	Plateau Mean	Delta
Dense regular	0.78	0.50–0.10	0.89	+0.11
Dense adv.-heavy	0.61	0.50–0.05	0.74	+0.13
Larger world	0.82	0.50	0.86	+0.04
Large world + clutter	0.68	0.50–0.10	0.86	+0.18
Large world + clutter (adv.)	0.61	0.50–0.10	0.79	+0.18

Three patterns emerge. First, under dense overload, pruning *improves* over the unpruned baseline — it acts as interference reduction, not just compression. Second, in semantically rich worlds with little redundancy, the safe plateau narrows. Third, when scale and clutter combine, interference reduction again dominates and the plateau widens. Collapse appears only when support is almost entirely stripped away. Bounded episodic memory is not merely tolerable but actively beneficial under dense overload.

5. Qualitative Results

We illustrate the quantitative findings with a representative adversarial revision case.

Example 1 — Memory Type Decomposition

Consider the query:

> Ready the Beacon update factoring in adjustments and the active repository line.

This requires recovering the current Beacon branch, the default structural preference, the Beacon-specific override, and supporting evidence for the default. `episodes_only` retrieves the branch and override but misses the default trait. `abstract_only` recovers the default but misses the current fact and override. Neither alone is sufficient.

Example 2 — Flat Mixed vs. Hierarchical Retrieval

On the same case, `mixed_flat` recovers the branch, default trait, and override — more than either baseline — but still misses default-support evidence. The artifact is plausible but incomplete under strict scoring. Hierarchical `mixed` succeeds on all fields, recovering the supporting child evidence attached to the default abstraction. This is the piece that raises the

artifact from partially correct to exactly correct. The pattern is stable: `mixed_flat` characteristically loses support evidence that does not rank highly on query similarity but is reachable through abstraction-first child descent.

These examples illustrate a consistent mechanistic pattern: as task ambiguity increases, the system progressively requires recall (episodic), generalization (abstract), conflict-aware selection (revision), and provenance-sensitive disambiguation (adversarial revision). Hierarchy becomes critical precisely where these demands compound.

---

6. Related Work

6.1 Memory-Augmented Neural Networks

Memory-augmented architectures (Graves et al., 2014; Weston et al., 2015; Sukhbaatar et al., 2015) extend neural networks with external stores accessed via learned read/write operations, typically storing and retrieving vectors attended over uniformly. EHM differs in that memory nodes carry structured semantic content organized in an explicit graph, and inference is conditioned on the *type* and *hierarchical position* of retrieved nodes rather than on uniform attention over a flat memory bank.

6.2 Retrieval-Augmented Generation

RAG systems (Lewis et al., 2020; Borgeaud et al., 2022) treat retrieved items as interchangeable context chunks. EHM's contribution is functional decomposition: the system distinguishes episodic grounding from abstract structure and applies different inference mechanisms depending on conflict type. Hierarchical organization supports compositional queries that flat retrieval cannot express — a query may require both a specific episode and the abstraction that governs it.

6.3 Hierarchical and Structured Memory

Hierarchical memory has been explored in reinforcement learning (Lampinen et al., 2021) and cognitive architectures (Laird, 2012). EHM organizes explicit memory nodes in a graph and conditions inference on their structural relationships. The hierarchy is over *experience* — episodes grouped under abstractions that enable generalization — not over parameters or weight-space updates.

6.4 Looped Language Models

LoopLM (Zhu et al., 2025) applies parameter-shared transformer blocks recurrently, improving knowledge *manipulation* without expanding *storage* capacity. This directly motivates external structured memory: if parametric storage is bounded regardless of architecture, systems requiring long-horizon reasoning over accumulated experience must look outside parametric memory.

6.5 Sparse Routing and Mixture-of-Experts

Mixture-of-Experts models (Shazeer et al., 2017; Fedus et al., 2022) establish that routing inputs to sparse subsets is feasible and beneficial. EHM shares the selective-activation principle but routes over structured memory nodes with typed relationships rather than over interchangeable parameter blocks.

6.6 Adapter-Based Memory and Trainable Memory Modules

Recent work treats LoRA as a memory mechanism (Back et al., 2025), explores adapter-based memory pipelines (Bini et al., 2025), and proposes trainable external memory at scale (Wei et al., 2025; Berges et al., 2024). EHM differs in that memory remains fully explicit and hierarchically organized, with the model accessing memory through typed tools or structured retrieval rather than adapter-mediated absorption. The tool-mediated LLM result (Section 4.4) shows explicit access succeeding where implicit conditioning failed; the adapter literature suggests a learned reader could further improve the interface.

7. Discussion

7.1 Summary of Claims

The results support four nested claims. First, episodic and abstract memory are functionally distinct: each is necessary for a different class of task, and neither subsumes the other. Second, having both memory types is necessary but not sufficient for revision: flat mixed retrieval fails on the hardest adversarial cases because it loses the supporting evidence structure. Third, hierarchical retrieval — specifically abstraction-first child descent — substantially improves structured downstream revision outputs beyond both single-memory baselines and flat mixed retrieval. Fourth, the hierarchy advantage extends beyond deterministic artifact construction: it survives into tool-mediated LLM generation (most clearly on adversarial cases) and remains robust under staged episodic pruning across diverse memory-load regimes.

7.2 The Reasoning–Generation Boundary

The deterministic downstream benchmark isolates the memory architecture's contribution by removing generation quality as a confound. The tool-mediated LLM result (Section 4.4) provides initial evidence that the hierarchy advantage can cross the reasoning–generation boundary, but with important caveats. The LLM-in-the-loop hierarchy gain is concentrated on adversarial cases requiring conflict resolution and provenance-sensitive integration; on easier revision cases, the LLM layer introduces synthesis noise that narrows the gap between flat and hierarchical retrieval. The positive LLM result comes specifically from explicit tool-mediated memory access — earlier attempts using rendered text briefs, latent prefix conditioning, and adapter-based readers all failed to preserve the hierarchy signal. This suggests that the memory–generation interface is a key design dimension: the architecture of the interface matters as much as the architecture of the memory itself. Whether the hierarchy advantage extends to fully unconstrained free-form generation remains an open question that will likely require a stronger model-side reader (Section 8).

7.3 Memory Budgets and Pruning

The pruning results (Section 4.5) address a practical concern for any system that accumulates experience over time. Under dense memory overload, staged episodic pruning can act as both compression and interference reduction, sometimes *improving* over the full unpruned store. The effect is most pronounced when dense clutter creates retrieval interference, and least pronounced when the memory store is semantically rich with little redundancy. This result supports the viability of bounded-memory operation for EHM and identifies interference reduction as a side benefit of memory management, not merely a cost to be tolerated.

7.4 Limitations

Several limitations bound the current claims:

Synthetic evaluation only. All experiments use controlled synthetic benchmarks. The case families, entity distributions, and interference patterns are generated rather than drawn from natural task distributions. We regard the synthetic design as appropriate for isolating memory mechanisms, but generalization to naturalistic settings is not yet established.

Initial LLM-in-the-loop scale. The tool-mediated LLM result is positive but remains at dev scale (8 seeds, 4 cases per type per seed) with a single small instruct model. The deterministic 10-seed matrix is more stable and remains the primary quantitative centerpiece. Whether the hierarchy advantage grows or shrinks with larger models, longer generation budgets, and more diverse task families is an empirical question.

Rule-based routing. The routing policy is lightweight and rule-based. The current results are therefore conservative with respect to routing: gains from hierarchy are achieved despite unsophisticated routing, and a learned router might amplify or diminish these gains.

Simple consolidation. The abstraction-building mechanism is fixed cosine-threshold clustering over deterministic latent vectors. This is sufficient to demonstrate the value of hierarchical retrieval, but a learned consolidation mechanism could produce higher-quality abstractions.

Scale. The benchmark operates at synthetic-task scale with modest memory banks. The pruning experiments begin to probe larger memory stores, but whether the hierarchy advantage persists as the memory store grows by orders of magnitude is an empirical question for future work.

8. Conclusion and Future Work

We present a hierarchical memory architecture designed to support long-horizon reasoning. In the current vector-first implementation, episodic and abstract memories are functionally distinct, and hierarchical retrieval improves structured downstream revision outputs beyond both single-memory baselines and flat mixed retrieval in initial synthetic testing, demonstrating inherent value in hierarchical organization. The hierarchy advantage extends into tool-mediated LLM generation, with the clearest gains on adversarial cases requiring conflict resolution and provenance-sensitive integration. Staged episodic pruning indicates that bounded-memory operation is viable and can improve over unpruned baselines under dense overload. Together, these results provide a foundation for future systems integrating structured hierarchical memory with learned generation.

Future Directions

The next steps divide naturally into two groups: the direct continuation of the current architecture, and more orthogonal extensions that could add value once that continuation is in place.

Linear continuation of the current system. The most direct research path from the present results is:

1. **Learned consolidation.** The current abstraction layer is formed by fixed cosine-threshold clustering over deterministic latent vectors. A natural next step is to replace this with a learned consolidation mechanism that decides what should be grouped, what should remain episodic, and what structure should be preserved in the abstract node. Adapting

richer, non-centroid structures for abstraction — enabling de-consolidation where useful — is a complementary direction.

2. **Learned routing.** The current router is deliberately lightweight and rule-based. Once consolidation becomes learned, the next step is to learn routing as well: adapting the depth and breadth of retrieval for a given query to maximize efficiency and efficacy of memory utilization.
3. **Direct latent conditioning.** Storage, consolidation, routing, and retrieval are already vector-first, but the final conditioning boundary is still rendered into a structured textual view for the downstream task. The next architectural step is to condition learned models directly on latent memory objects or latent memory summaries, reducing the remaining dependence on compatibility text. Recent work on trainable memory modules (Wei et al., 2025; Berges et al., 2024) and adapter-based memory systems (Bini et al., 2025) suggests that stronger model-side readers are now plausible, and that attention-only or tiny-adapter designs are likely too weak (Back et al., 2025). The tool-mediated LLM result (Section 4.4) provides a complementary data point: explicit cognitive memory tools already preserve the hierarchy advantage, suggesting that a trainable reader should target deeper integration rather than minimal prefix conditioning.
4. **Scaled neural downstream generation.** The tool-mediated LLM result (Section 4.4) provides initial evidence that the hierarchy advantage can cross the reasoning–generation boundary. The next step is to scale this evaluation to larger seed matrices, larger models, and more diverse task families, and to compare explicit tool-mediated memory access with learned latent conditioning to identify which interface design best preserves the hierarchy signal at scale.

These four steps form the clearest linear continuation of the present work: learned memory formation, learned memory access, learned memory conditioning, and learned output realization.

Orthogonal extensions. Several promising directions are less central to that straight-line progression but could add substantial value. Here are two:

Combining hierarchical memory with iterative latent computation. LoopLM (Zhu et al., 2025) improves knowledge manipulation without expanding storage capacity. A natural extension is combining a looped base model with memory-conditioned context from EHM, yielding a system that benefits from both structured knowledge expansion and enhanced compositional reasoning. A key empirical question is whether iterative refinement can learn to differentially weight episodic versus abstract contributions across loop steps, performing selection as an emergent property of latent recurrence.

Interaction learning. The current architecture treats the memory bank as static at inference

time. Another extension is to close the loop between inference and memory formation — using interaction traces and feedback signals to refine existing memories and propose new ones, making the memory graph an evolving experiential substrate rather than a fixed artifact of offline construction.

Finally, our current architecture complements existing pre-trained models by adding explicit hierarchical memory as an external system. Whether hierarchical relationships could be derived from existing model weights, or incorporated directly into pre-training to allow models to form and maintain memory hierarchies as a native capability, is an open question. The results presented here suggest that hierarchical structure carries value independent of substrate, motivating exploration of hierarchy-aware training objectives alongside the external-memory approach.

References

Back, Seungyeon, et al. 2025. *Understanding LoRA As Knowledge Memory: An Empirical Analysis*. OpenReview. https://openreview.net/forum?id=i1Mi2R1TsU

Berges, Vincent-Pierre, et al. 2024. *Memory Layers at Scale*. arXiv. https://arxiv.org/abs/2412.09764

Bini, Tommaso, et al. 2025. *MemLoRA: Distilling Expert Adapters for On-Device Memory Systems*. arXiv. https://arxiv.org/abs/2512.04763

Borgeaud, Sebastian, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, et al. 2022. *Improving Language Models by Retrieving from Trillions of Tokens*. arXiv. https://arxiv.org/abs/2112.04426

De Cao, Nicola, Wilker Aziz, and Ivan Titov. 2021. *Editing Factual Knowledge in Language Models*. EMNLP 2021. https://aclanthology.org/2021.emnlp-main.522/

Edge, Darren, Haishi Li, Huda Sharaf, Claire Zhang, and Alison Halevy. 2024. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. Microsoft Research. https://www.microsoft.com/en-us/research/publication/from-local-to-global-a-graph-rag-approach-to-query-focused-summarization/

Fedus, William, Barret Zoph, and Noam Shazeer. 2022. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. Journal of Machine Learning Research 23(120). https://jmlr.org/papers/v23/21-0998.html

Graves, Alex, Greg Wayne, and Ivo Danihelka. 2014. *Neural Turing Machines*. arXiv. https://arxiv.org/abs/1410.5401

Gutiérrez, Bernal Jiménez, et al. 2024. *HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models*. arXiv. https://arxiv.org/abs/2405.14831

Hendrycks, Dan, et al. 2025. *A definition of AGI.* arXiv preprint arXiv:2510.18212. https://arxiv.org/abs/2510.18212

Laird, John E. 2012. *The Soar Cognitive Architecture*. MIT Press. https://mitpress.mit.edu/9780262122965/the-soar-cognitive-architecture/

Lampinen, Andrew Kyle, Stephanie C. Y. Chan, Andrea Banino, and Felix Hill. 2021. *Towards Mental Time Travel: A Hierarchical Memory for Reinforcement Learning Agents*. NeurIPS 2021. https://proceedings.neurips.cc/paper/2021/hash/ed519dacc89b2bead3f453b0b05a4a8b-Abstract.html

Lewis, Patrick, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. 2020. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS 2020. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Meng, Kevin, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. *Mass-Editing Memory in a Transformer*. ICLR 2023. https://arxiv.org/abs/2210.07229

Mitchell, Eric, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. *Fast Model Editing at Scale*. ICLR 2022. https://arxiv.org/abs/2110.11309

Riley, Chris. 2026. *Abstraction and Resilience: the Characteristics of Consciousness*. Smashwords. https://www.smashwords.com/books/view/2000838

Sarthi, Parth, Salman Abdullah, Ian Frosst, and Mohammad Alizadeh. 2024. *RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval*. arXiv. https://arxiv.org/abs/2401.18059

Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. ICLR 2017. https://openreview.net/forum?id=B1ckMDqlg

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. *End-To-End Memory Networks*. NeurIPS 2015. https://arxiv.org/abs/1503.08895

Wei, Jason, et al. 2025. *MLP Memory: Language Modeling with Retriever-pretrained External Memory*. arXiv. https://arxiv.org/abs/2508.01832

Weston, Jason, Sumit Chopra, and Antoine Bordes. 2015. *Memory Networks*. ICLR 2015. https://arxiv.org/abs/1410.3916

Zhu, Rui-Jie, et al. 2025. *Scaling Latent Reasoning via Looped Language Models*. arXiv. https://arxiv.org/abs/2510.25741