TOWARD A THEORY OF HIERARCHICAL MEMORY FOR LANGUAGE AGENTS

Yashar Talebirad
University of Alberta
talebira@ualberta.ca

Ali Parsaee
University of Alberta
Parsaee@ualberta.ca

Csongor Y. Szepesvari
University of Alberta
csongor@ualberta.ca

Amirhossein Nadiri
York University
anadiri@yorku.ca

Osmar Zaiane
University of Alberta
zaiane@ualberta.ca

ABSTRACT

Many recent long-context and agentic systems address context-length limitations by adding *hierarchical memory*: they extract atomic units from raw data, build multi-level representatives by grouping and compression, and traverse this structure to retrieve content under a token budget. Despite recurring implementations, there is no shared formalism for comparing design choices. We propose a unifying theory in terms of three operators. **Extraction** (α) maps raw data to atomic information units; **coarsening** ($C = (\pi, \rho)$) partitions units and assigns a representative to each group; and **traversal** (τ) selects which units to include in context given a query and budget. We identify a **self-sufficiency** spectrum for the representative function ρ and show how it constrains viable retrieval strategies (a **coarsening-traversal coupling**). Finally, we instantiate the decomposition on eleven existing systems spanning document hierarchies, conversational memory, and agent execution traces, showcasing its generality.

1 INTRODUCTION

Language agents increasingly rely on large external corpora and long interaction histories, but larger context windows do not reliably improve information use. Models underweight mid-context evidence (Liu et al., 2024), struggle with single-fact retrieval in long documents (Hsieh et al., 2024), and often degrade as context grows ("context dilution"/"context rot"). This reveals a gap between *capacity* (more tokens) and *control* (what to surface, and when). Hierarchical memory is a recurring response: RAPTOR (Sarthi et al., 2024), GraphRAG (Edge et al., 2024), xMemory (Hu et al., 2026), H-MEM (Sun & Zeng, 2025), and SimpleMem (Liu et al., 2026) build multi-level representations by grouping, compressing, and selectively expanding under a budget; analogous ideas appear in agent execution-trace memory, including reasoning-focused traces (MemoBrain (Qian et al., 2026), StackPlanner (Zhang et al., 2026)) and action/state traces (InfiAgent (Yu et al., 2026a)). Yet these systems are usually presented as isolated design points, making it hard to compare assumptions.

We show that, every hierarchical memory system, whether it operates on stored knowledge or live agent execution traces, instantiates a single three-operator pipeline (Figure 1). Extraction (α) maps raw data to a graph of atomic information units. Coarsening ($C = (\pi, \rho)$) partitions units into groups and produces a representative for each group, yielding a smaller graph, and iterating this produces the hierarchy. Traversal (τ) takes the hierarchy, a query, and token budget, and returns a subset of atomic units to include in context. The central theoretical observation is that the representative function ρ varies along a self-sufficiency spectrum: a detailed summary preserves most of the group's information, while a category label preserves almost none. This property constrains which retrieval strategy works. Self-sufficient representatives support collapsed search over all levels (RAPTOR), while referential representatives require top-down refinement (H-MEM, xMemory). We call this constraint the coarsening-traversal (C–T) coupling.

Prior efforts classify memory systems but do not formalize hierarchy construction. CoALA (Sumers et al., 2024) organizes memory by cognitive type (working, episodic, semantic, procedural), and

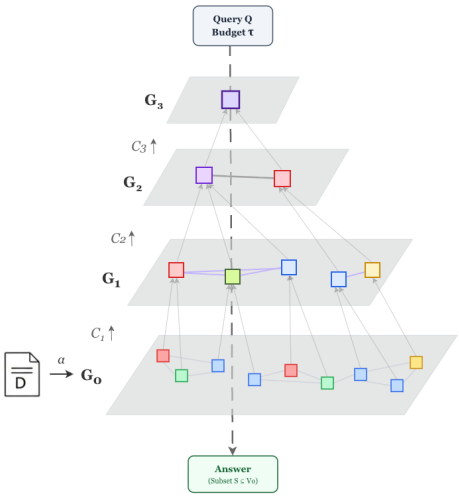

Figure 1: The (α, C, τ) pipeline: D is extracted (α) into atoms G_0; coarsening C_1, C_2, C_3 yields layers G_1, G_2, G_3; traversal τ takes a query and budget and returns a subset S of atoms.

MemEngine (Zhang et al., 2025b) implements multiple systems from shared primitives. We provide a mathematical unification, extended to agent execution traces: a formal (α, C, τ) decomposition with information monotonicity, a self-sufficiency spectrum with C–T coupling, and an instantiation of this decomposition across eleven data and trace systems.

2 THE FRAMEWORK

2.1 CORE DEFINITIONS

Natural language exhibits multiscale structure (themes, topics, individual facts), and queries may target any scale. We define three operators that build multiresolution representations of this structure.

Information units. Let $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_p$ be a product of feature spaces with one distinguished factor $\mathcal{F}_c = \Sigma^*$ (text). A *unit* $u \in \mathcal{F}$ has projections $\phi_i(u)$; $\phi_c(u)$ is the content. Typical attributes include embeddings ϕ_{emb}, timestamps ϕ_{time}, entities ϕ_{ent}, and structural paths ϕ_{path}.

Unit graph. $G = (U, E)$ with U a finite set of units and $E \subseteq U \times U$. The edge set may be empty (a flat bag of units, as in SimpleMem), may encode entity-entity relationships (a knowledge graph, as in GraphRAG), or may encode sequential adjacency (a conversation stream, as in xMemory).

Extraction. The first operator maps raw data into structured units. $\alpha : \mathcal{D} \to \mathcal{G}$ maps raw data D to a unit graph $G_0 = (U_0, E_0)$. All preprocessing (chunking, named entity recognition, path assignment) happens inside α, and thus, each unit is self-contained in the resulting graph G_0.

Coarsening. A coarsening operator on $G = (U, E)$ is $C = (\pi, \rho)$: (i) $\pi : U \twoheadrightarrow [m]$ is a surjective *grouping* that induces the partition $\{G_j = \pi^{-1}(j)\}$; (ii) $\rho : 2^U \to \mathcal{F}$ is a *representative function* such that $\rho(G_j) \in \mathcal{F}$. We assume $|U'| = m < |U|$, where $U' = \{\rho(G_j) : j \in [m]\}$.[1]

Hierarchy. Iterating C_i produces a hierarchy $\mathcal{H} = (G_0, C_1, \ldots, C_L)$ with $G_\ell = C_\ell(G_{\ell-1})$.[2]

Denote the node set of G_ℓ as V_ℓ. Level 0 is thus the atom set V_0; each application of C_ℓ builds the next layer V_ℓ from $V_{\ell-1}$ by grouping nodes (via π_ℓ) and assigning one representative per group (via ρ_ℓ), with edges from each representative to its children. The hierarchy can thus be viewed as a multi-layer graph: the bottom layer holds fine-grained units, and each higher layer is a coarsened

[1] Some systems (RAPTOR's GMM soft clustering, overlapping community detection) allow a unit to belong to multiple groups, replacing $\pi : U \to [m]$ with $\pi : U \to 2^{[m]} \setminus \emptyset$; the hierarchy then becomes a DAG.

[2] $\mathcal{H}$ lists the initial graph G_0 and the coarsening *operators*; the graphs $G_1, \ldots, G_L$ are the *results* of applying them, not separate entries in the tuple. The "layers" in the multi-level view are these derived graphs.

graph whose nodes summarize the subgraph below (Figure 1). Each C_ℓ may use a different grouping and representative strategy (e.g., xMemory uses temporal grouping at level 1, semantic grouping at level 2, and thematic grouping at level 3).

2.2 Multi-Resolution Representation

Assume the input data D is drawn from a distribution over $\mathcal{D}$. Since $V_0 = \alpha(D)$ and each $V_\ell = C_\ell(V_{\ell-1})$, every level inherits randomness from D, and the entropies below are well defined. Let $\mathcal{H} = (G_0, C_1, \ldots, C_L)$, and assume each C_ℓ is deterministic and information-losing with positive probability under D (equivalently, not almost surely invertible). Then: (i) $H(V_0) > H(V_1) > \cdots > H(V_L)$, since determinism gives $H(V_\ell) \leq H(V_{\ell-1})$ and strict information loss on a nonzero-probability set makes the inequality strict; (ii) for any query variable Q with query-agnostic coarsening (i.e., V_ℓ depends only on $V_{\ell-1}$ and not on Q), $I(Q; V_0) \geq I(Q; V_1) \geq \cdots \geq I(Q; V_L)$, since learning V_ℓ provides no additional information about Q once $V_{\ell-1}$ is known; equivalently, Q–$V_{\ell-1}$–V_ℓ forms a Markov chain (and the Data Processing Inequality applies).

This formalizes that every coarsening step loses information.[3] The atoms V_0 set the information ceiling; finer extraction increases $H(V_0)$ and raises the ceiling on downstream retrieval quality.[4]

2.3 Self-Sufficiency and the C–T Coupling

The representative function ρ is where systems diverge most, and its properties determine the behavior of the entire hierarchy. The **self-sufficiency** of ρ at group G_j measures how much of the group's information the representative preserves: [5]

$$\mathrm{SS}(\rho, G_j) = \frac{I(G_j; \rho(G_j))}{H(G_j)} = 1 - \frac{H(G_j \mid \rho(G_j))}{H(G_j)}. \tag{1}$$

Assume $H(G_j) > 0$; if $H(G_j) = 0$, define $\mathrm{SS} = 1$. When self-sufficiency is high (close to 1), the representative preserves most of the group's content and can answer queries on its own (e.g., LLM-generated summaries). When self-sufficiency is low (close to 0), the representative serves only as a routing label, indicating what the group contains without reproducing it (e.g., domain names). For any query Q, $I(Q; G_j) - I(Q; \rho(G_j)) \leq H(G_j \mid \rho(G_j))$, so high self-sufficiency bounds the information loss incurred by reading the representative rather than the children.

C–T coupling. Self-sufficiency constrains which traversal strategies are viable, reflecting a rate-distortion tradeoff. Self-sufficient ρ operates at low distortion: coarse nodes carry enough information to answer queries directly, so collapsed search suffices (single-shot decoding; RAPTOR). Referential ρ operates at high distortion: coarse nodes serve only as routing labels, so top-down refinement is needed to recover the lost detail (successive refinement; H-MEM, xMemory). Mismatching ρ and traversal wastes budget: collapsed search over referential representatives spends tokens on uninformative labels, while top-down routing through self-sufficient representatives spends tokens on unnecessary expansion. Existing evidence is consistent with this observation, but a controlled comparison across the spectrum remains open.

2.4 Traversal

A **traversal** $\tau : (\mathcal{H}, q, B) \to S \subseteq V_0$ takes a hierarchy, query, and token budget and returns atoms satisfying $\sum_{u \in S} |\phi_c(u)| \leq B$. By pruning via hierarchy structure, traversal can reduce relevance evaluations from $O(n)$ (flat search) to $O(\log n)$ under bounded compute. Four patterns recur: top-down refinement (H-MEM, xMemory), collapsed search across levels (RAPTOR), multi-view parallel retrieval (SimpleMem), and reasoning-based navigation (PageIndex (Zhang et al., 2025a), MemWalker (Chen et al., 2023)).[6] This reflects a dual role of $\mathcal{H}$: with referential ρ, it mainly serves

[3] This also aligns with the Information Bottleneck principle (Tishby et al., 1999): as compression becomes more aggressive at each level of $\mathcal{H}$, representations move along a trade-off curve between retained detail and compactness, suggesting a multi-resolution structure.

[4] Appendix B relates data size, context window, and compression ratio to the number of levels needed.

[5] A computable LLM-based SS_θ and a query-dependent refinement SS_Q are developed in Appendix A.

[6] Pseudocode appears in Appendix D.

Table 1: Data and agent execution-trace systems as (α, C, τ). Type: D = data, T = trace.

System	Type	α	π; ρ (SS)	τ
RAPTOR	D	100-tok chunks	UMAP+GMM; LLM summary (high)	collapsed / top-down
GraphRAG	D	entity/rel extract	Leiden; community report (high)	global / entity fan-out
xMemory	D	episode boundaries	guided split/merge; fact distill (mid)	top-down + expand
H-MEM	D	episode + profile	LLM 4-level; domain labels (low)	top-down FAISS
SimpleMem	D	window + density	online synthesis; consolidated (high)	multi-view parallel
Mastra's OM	D	token threshold	all-in-one; Reflector (high)	full log
PageIndex	D	structural parse	doc structure; titles/summaries (mid)	reasoning-based
MemoBrain	T	episode $\rightarrow$ thought	dep.-subgraph; Fold (high) / Flush (low)	context projection
StackPlanner	T	task stack entries	segment; Condensation (high) / Pruning (low)	stack top + retrieval
AgeMem	T	turns $\rightarrow$ entries	learned; SUMMARY (high) / FILTER (low)	RETRIEVE + FILTER
InfiAgent	T	artifacts $\rightarrow$ files	state consolidation (mid)	bounded $g(\mathcal{F}_t, a_{t-k:t-1})$

as an *index* for top-down routing; with self-sufficient ρ, it also serves as a *representation* whose non-leaf nodes often answer broad queries directly (collapsed search).

3 Instantiation

3.1 Data and Trace Systems

To showcase generality, Table 1 maps eleven systems to (α, C, τ). **Data memory** systems operate on stored content: batch document hierarchies (RAPTOR, GraphRAG), online conversational memory (xMemory, H-MEM, SimpleMem), observational compression (Mastra's Observational Memory (Barnes, 2026)), and structured navigation (PageIndex). **Agent execution-trace** systems operate on execution: α segments traces, C groups/compresses them (e.g., MemoBrain's Fold/Flush, StackPlanner's Condensation/Pruning), and τ reconstructs working context. In trace memory, coherence is causal-functional (steps for the same subproblem) rather than semantic, and the training signal is task reward (DPO, GRPO) rather than retrieval metrics. AgeMem (Yu et al., 2026b) exposes store/retrieve/summarize/filter/discard as tool actions, while InfiAgent uses bounded reconstruction from workspace state $\mathcal{F}_t$ and recent actions (Yu et al., 2026a).[7]

4 Discussion and Future Work

The (α, C, τ) decomposition provides a common language for comparing systems. Table 1 shows that data-memory and agent-trace systems converge on the same pipeline despite different motivations. The C–T coupling suggests that ρ and τ should be chosen jointly, and the Fano bound (Appendix B) quantifies how representative quality caps branching factor. Current grouping functions π mostly assume unweighted, deterministic edges, but real unit graphs are often weighted or uncertain. Local community mining and search methods for weighted (SIWOw (Zafarmand et al., 2023)) and uncertain (USIWO (Talebirad et al., 2023)) graphs motivate extending π to exploit edge strength/confidence and improve partition coherence (Appendix C).

As agent execution traces grow, most prior steps become irrelevant to the current subtask, creating a similar context pressure that motivates hierarchical organization of stored knowledge. This convergence suggests that complex agentic tasks involve three coupled hierarchies: the data hierarchy $\mathcal{H}_{\text{data}}$ (stored knowledge), the trace hierarchy $\mathcal{H}_{\text{trace}}$ (reasoning, interaction, and action traces), and the task hierarchy $\mathcal{H}_{\text{task}}$ (problem decomposition). Each node in $\mathcal{H}_{\text{task}}$ induces a traversal through $\mathcal{H}_{\text{data}}$ and generates nodes in $\mathcal{H}_{\text{trace}}$; the inter-hierarchy alignment constraints remain open.

The central limitation is the assumption of a static hierarchy. In practice, hierarchies evolve via insertion/restructuring (xMemory's split/merge, MemTree (Rezazadeh et al., 2025)), and retrieval can reshape memory by reinforcement or decay. Both dynamics break the Markov-chain argument of Section 2.2: query-conditioned ρ makes V_ℓ depend on Q, and stateful coarsening couples τ and C into a feedback loop. Formalizing this setting (likely via adaptive information theory or online learning) is the most pressing open problem. Empirical next steps are validating SS_θ as a predictor of retrieval quality, testing C–T coupling with matched ρ/τ variants, and computing IB-optimal ρ.

[7] Bounded context forces a scope–nuance trade-off, which explains convergence on "capture at full granularity, coarsen iteratively, reconstruct on demand via τ".

REFERENCES

Tyler Barnes. Observational memory: 95% on longmemeval. Mastra Research Blog, February 2026. URL https://mastra.ai/research/observational-memory. Observational Memory (OM).

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023. MemWalker; Meta AI Research.

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024. arXiv:2404.06654.

Zhanghao Hu, Qinglin Zhu, Hanqi Yan, Yulan He, and Lin Gui. Beyond RAG for agent memory: Retrieval by decoupling and aggregation. *arXiv preprint arXiv:2602.02007*, 2026.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents. *arXiv preprint arXiv:2601.02553*, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. arXiv:2307.03172.

Hongjin Qian, Zhao Cao, and Zheng Liu. Memobrain: Executive memory as an agentic brain for reasoning. *arXiv preprint arXiv:2601.08079*, 2026.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *International Conference on Learning Representations*, 2025. MemTree; arXiv:2410.14052.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. arXiv:2309.02427.

Haoran Sun and Shaoning Zeng. Hierarchical memory for high-efficiency long-term reasoning in LLM agents. *arXiv preprint arXiv:2507.22925*, 2025.

Yashar Talebirad, Mohammadmahdi Zafarmand, Osmar R. Zaiane, and Christine Largeron. USIWO: A local community search algorithm for uncertain graphs. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pp. 187–194, 2023.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, 1999.

Chenglin Yu, Yuchen Wang, Songmiao Wang, Hongxia Yang, and Ming Li. Infiagent: An infinite-horizon framework for general-purpose autonomous agents. *arXiv preprint arXiv:2601.03204*, 2026a.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026b. AgeMem.

Mohammadmahdi Zafarmand, Yashar Talebirad, Eric Austin, Christine Largeron, and Osmar R. Zaïane. Fast local community discovery relying on the strength of links. *Social Network Analysis and Mining*, 13(1):112, 2023.

Mingtian Zhang, Yu Tang, and PageIndex Team. Pageindex: Next-generation vectorless, reasoning-based rag. *PageIndex Blog*, September 2025a. https://pageindex.ai/blog/pageindex-intro.

Ruizhe Zhang, Xinke Jiang, Zhibang Yang, Zhixin Zhang, Jiaran Gao, Yuzhen Xiao, Hongbin Lai, Xu Chu, Junfeng Zhao, and Yasha Wang. StackPlanner: A centralized hierarchical multi-agent system with task-experience memory management. *arXiv preprint arXiv:2601.05890*, 2026.

Zeyu Zhang, Quanyu Dai, Xu Chen, Rui Li, Zhongyang Li, and Zhenhua Dong. Memengine: A unified and modular library for developing advanced memory of LLM-based agents. *arXiv preprint arXiv:2505.02099*, 2025b.

A Alternate Self-Sufficiency Measures

To make the C–T coupling testable in practice, we need proxies for self-sufficiency that can be estimated on real systems. The measures below separate what can be assessed at construction time from what is revealed only under concrete queries.

Computable proxy (SS_θ). Shannon entropy is not directly computable for natural language, so we use an LLM P_θ as a practical proxy:

$$\mathrm{SS}_\theta(\rho, G_j) \;=\; 1 - \frac{-\log P_\theta(G_j \mid \rho(G_j))}{-\log P_\theta(G_j)}. \tag{2}$$

Intuitively, this measures how much the representative helps the model predict the group's contents. Since many systems use LLMs to generate representatives, SS_θ directly affects downstream quality. If ρ hallucinates (adds facts not supported by the children), then $P_\theta(G_j \mid \rho)$ drops: the representative no longer predicts the actual children and may actively mislead. The same drop occurs if ρ blurs important distinctions (e.g., by merging dissimilar items into a vague summary), because the representative fails to disambiguate.

These failure modes suggest that SS_θ is a reasonable signal of the reliability of ρ. The decision rule linking SS to traversal choice (Section 2.3) holds only when SS_θ indicates that ρ is reliable. When SS_θ is low (due to aggressive compression, hallucination, or incoherent grouping), the system should default to treating ρ as referential, regardless of the nominal compression ratio.

Query-dependent refinement (SS_Q). Both SS and SS_θ are query-independent: they measure the fraction of a group's total information preserved by ρ. A query Q may need only a small portion of G_j's information. The **query-dependent self-sufficiency**

$$\mathrm{SS}_Q(\rho, G_j, Q) \;=\; \frac{I(Q;\, \rho(G_j))}{I(Q;\, G_j)} \tag{3}$$

measures the fraction of query-relevant information preserved by $\rho(G_j)$ when $I(Q; G_j) > 0$ (if $I(Q; G_j) = 0$, define $\mathrm{SS}_Q = 1$). Since $\rho(G_j)$ is a deterministic function of G_j, the chain Q–G_j–$\rho(G_j)$ is Markov and the Data Processing Inequality gives $\mathrm{SS}_Q \in [0, 1]$.

Define query relevance $r_Q = I(Q; G_j)/H(G_j)$. From $I(Q; G_j \mid \rho(G_j)) \le H(G_j \mid \rho(G_j))$ and the identity $\mathrm{SS}_Q = 1 - I(Q; G_j \mid \rho(G_j)) \,/\, I(Q; G_j)$:

$$\mathrm{SS}_Q \;\ge\; 1 - \frac{1 - \mathrm{SS}}{r_Q}. \tag{4}$$

When r_Q is large, SS is a tight lower bound on SS_Q. When r_Q is small, the bound does not provide a meaningful constraint, and low-SS representatives can still achieve high SS_Q. This explains why referential representatives (category labels, hash keys, B-tree boundary keys) are effective for routing: the routing query requires very little of the group's total information ($r_Q \approx 0$), so even $\mathrm{SS} \approx 0$ is compatible with $\mathrm{SS}_Q \approx 1$.

SS is the appropriate construction-time metric, since no query is available when ρ is built. At design time, SS determines the default traversal strategy via the C–T coupling (Section 2.3). At query time, SS_Q determines whether the representative suffices or the children must be expanded. Operationally, SS is a prior over expected query-time behavior and should be calibrated on a target workload by comparing SS against empirical means of SS_Q. Accordingly, the C–T coupling is a practical default: high SS suggests more queries are answerable from the representative (collapsed search), while low SS suggests more queries require expansion (top-down refinement).

B Fano Bound and Branching Factor

The Fano inequality constrains how many groups a level can contain given the discriminative quality of the representative.

Theorem (Fano). Let $Z \in \{1, \ldots, n_k\}$ be the correct group index and O the observable routing evidence. For any estimator $\hat{Z} = g(O)$ with error probability $p_e = P[\hat{Z} \neq Z]$:

$$p_e \geq 1 - \frac{I(Z;O) + 1}{\log_2 n_k}.$$

Corollary (from xMemory (Hu et al., 2026)). If $I(Z;O) \leq B$ bits and $p_e \leq \varepsilon$, then $n_k \leq 2^{(B+1)/(1-\varepsilon)}$.

In our framework, B is determined by the information content of ρ: low-SS representatives (small B) force coarser partitions, while high-SS representatives permit finer ones. Composing across L levels with per-level error ε_ℓ, the total routing failure probability is bounded by $\sum_\ell \varepsilon_\ell$ (union bound).

Optimal branching. Assuming a balanced tree, uniform per-node cost, and no caching, top-down retrieval with beam width k and branching factor b has total cost $L \cdot k \cdot b$. With $n = |V_0| = b^L$, substituting $L = \ln n / \ln b$ yields cost $\propto b / \ln b$, which is minimized at $b = e$. The Fano constraint caps b at $2^{(B+1)/(1-\varepsilon)}$; if this is below e, the cap binds. The optimal depth is $L^* = \lceil \ln n / \ln b^* \rceil$.

Depth bound. A related structural constraint governs hierarchy depth. If total data comprises N tokens, the context window holds C tokens, and each coarsening level compresses content by a factor r, the top level fits in context only when $L \geq \lceil \log_r(N/C) \rceil$.

C Affinity and Coherence

Traversal quality depends not only on how informative representatives are, but also on whether grouping preserves meaningful neighborhood structure. Here, we make the hidden assumption behind top-down pruning more explicit: grouping must preserve local relevance geometry well enough that parent-level decisions remain predictive at child level. The C–T coupling (Section 2.3) concerns the representative ρ; the *grouping* π also constrains traversal. Every hierarchical system implicitly assumes that units placed in the same group are in some sense alike, whether by embedding similarity (RAPTOR), graph connectivity (GraphRAG), or structural path (PageIndex). Without a principled notion of "which units belong together," the partition is arbitrary and top-down pruning has no guarantee: a low-relevance parent might have highly relevant children in another topic that happened to be grouped with it. We formalize the notion of "alike" as an affinity and require that the partition respect it; that is what makes top-down traversal safe. The definitions below are an attempt to make this notion more precise.

Definition 1 (Affinity) *An affinity on a unit graph $G = (U, E)$ is a symmetric function $W : U \times U \to \mathbb{R}_{\geq 0}$.*

Definition 2 (W-coherent partition) *A partition $\pi : U \twoheadrightarrow [m]$ is W-coherent if within-group affinity exceeds between-group affinity:*

$$\mathbb{E}_{u,v:\pi(u)=\pi(v)}[W(u,v)] > \mathbb{E}_{u,v:\pi(u)\neq\pi(v)}[W(u,v)].$$

System	Affinity $W(u, v)$	Partition method
RAPTOR	$\cos(\phi_{\text{emb}}(u), \phi_{\text{emb}}(v))$	UMAP + GMM
GraphRAG	connectivity in E	Leiden community detection
H-MEM	LLM-judged domain co-membership	LLM 4-level classification
xMemory	cosine + sparsity-semantic score	guided split/merge
SimpleMem	semantic + temporal proximity	online LLM synthesis
Mastra's OM	temporal contiguity	token-count threshold
PageIndex	$\lvert\text{lcp}(\phi_{\text{path}}(u), \phi_{\text{path}}(v))\rvert$	structural parsing

Systems derive W from different signals:

In Table 1, Leiden community mining (used in GraphRAG) is the clearest example of global graph partitioning. The graph-mining literature also studies *community search*, which identifies a local dense subgraph around given query nodes without processing the full graph (Zafarmand et al., 2023). This suggests a local variant of π: rather than committing to a single global partition at construction time, the system could perform ad-hoc, on-demand coarsening around the regions of the unit graph most relevant to the current query. This way, the quality of π depends directly on the fidelity of the graph signals used by the mining step. Furthermore, real-world unit graphs frequently carry edge weights or probabilistic edges encoding uncertainty in the underlying facts, as commonly arises in knowledge-graph predicates with uncertain truth values. Incorporating such richer signals into W can tighten the coherence gap $\gamma_\ell = \mathbb{E}[W_{\text{within}}] - \mathbb{E}[W_{\text{between}}]$ and improve the quality of π.

The affinity W is why π should group certain units together, but π need not compute W explicitly. In structural parsing, π reads off path prefixes directly. For agent execution traces, a symmetric pairwise affinity is often a poor fit: MemoBrain groups reasoning thoughts by causal co-resolution (thoughts connected via directed dependencies that jointly resolve a subproblem), and StackPlanner groups stack entries by contiguous stack membership. In both cases, the grouping criterion is structural and directed rather than similarity-based, so we omit these systems from the table. The underlying principle still holds, however: W-coherence is what makes top-down traversal efficient, and its analogue in trace systems is that steps addressing the same subproblem should cluster, ensuring that parent relevance predicts child relevance. The key intuition is *relevance monotonicity*: if a parent node has low relevance to the query, its children are unlikely to have high relevance, with the gap controlled by the coherence gap γ_ℓ at that level. This justifies top-down pruning in H-MEM, xMemory, and MemWalker, and the same logic applies to MemoBrain's Fold and Flush: if a summarized sub-trajectory is irrelevant to the current reasoning state, its constituent steps are also likely irrelevant. Systems with poor coherence (e.g., noisy LLM partitions) should use wider beams or prefer collapsed search. A formal characterization of how γ_ℓ bounds child relevance conditioned on parent relevance remains open.

D Traversal Algorithms

The algorithms below operationalize the same design trade-off discussed in the main text: spend budget on broad routing first or on direct content evidence.

Algorithm 1: Top-down refinement (H-MEM, xMemory). Given $\mathcal{H}$, query q, budget B, beam widths $k_L, \ldots, k_0$: set $S_L \leftarrow \text{Top-}k_L(V_L, r(q, \cdot))$. For $\ell = L-1, \ldots, 0$: expand $\text{candidates} \leftarrow \bigcup_{v \in S_{\ell+1}} \text{children}(v)$, then $S_\ell \leftarrow \text{Top-}k_\ell(\text{candidates}, r(q, \cdot))$. Return S_0 truncated to budget B.

Algorithm 2: Collapsed search (RAPTOR). Pool all nodes across all levels: $\text{pool} \leftarrow \bigcup_{\ell=0}^{L} V_\ell$. Retrieve $S \leftarrow \text{Top-}k(\text{pool}, r(q, \cdot))$. Expand any non-leaf in S to its atoms. Return atoms within budget B.

Algorithm 3: Reasoning-based navigation (PageIndex, MemWalker). Initialize $\text{context} \leftarrow \phi_c(\text{root}(\mathcal{H}))$. While budget is not exhausted: the LLM selects a child node from context given q. If the node is a leaf, collect it. Otherwise, expand: $\text{context} \leftarrow \text{context} \cup \phi_c(\text{children}(\text{node}))$. Return collected atoms.

Algorithm 4: Multi-view parallel retrieval (SimpleMem). Decompose the query: $(q_{\text{sem}}, q_{\text{lex}}, q_{\text{sym}}, d) \leftarrow \text{Plan}(q)$. Set candidate limit $n \leftarrow f(d)$. Retrieve in parallel: $R_{\text{sem}} \leftarrow \text{Top-}n(V_0, r_{\text{sem}}(q_{\text{sem}}, \cdot))$, $R_{\text{lex}} \leftarrow \text{Top-}n(V_0, r_{\text{lex}}(q_{\text{lex}}, \cdot))$, $R_{\text{sym}} \leftarrow \text{Top-}n(V_0, r_{\text{sym}}(q_{\text{sym}}, \cdot))$. Return $R_{\text{sem}} \cup R_{\text{lex}} \cup R_{\text{sym}}$ truncated to budget B.

Algorithm 4 uses a flat multi-view design: each view indexes the same atoms V_0 using a different signal (semantic similarity, lexical overlap, symbolic metadata), and taking the union ensures that a relevant atom missed by one view can still be recovered by another. More generally, a system can maintain multiple hierarchies $\{\mathcal{H}_i\}$ over the same atom set V_0, each organizing atoms according to a different principle (e.g., one by temporal–thematic coherence, another by embedding similarity). This can be beneficial because any single grouping π is inevitably lossy: semantically related atoms may land in different branches, and the severity grows with partition granularity. A second hierarchy provides an alternative traversal path: an atom mis-partitioned in $\mathcal{H}_1$ may be correctly grouped in $\mathcal{H}_2$, so retrieval failure requires mis-partition in both views rather than just one. Multi-attribute queries benefit further because constraints of different types (e.g., topical relevance vs. temporal range) decompose naturally into per-hierarchy retrievals, which can be combined via set intersection for precision or union for recall; when constraints are independent, each retrieval can be parallelized.

E TASK AND AGENT HIERARCHIES

In Section 4, we mention three coupled hierarchies and note that $\mathcal{H}_{\text{task}}$ has dynamics different from data memory. A useful extension is to separate two orthogonal structures: the *task hierarchy* (what to solve) and the *agent hierarchy* (which solver tier handles each part).

Define $\mathcal{G}_{\text{task}} = (V_{\text{task}}, E_{\text{dep}})$ as a directed acyclic graph, which is gradually constructed during execution to represent dependencies among tasks; its hierarchical, coarse-to-fine ordering is denoted as $\mathcal{H}_{\text{task}}$. Let $(A, \leq_A)$ represent a set of agent tiers, partially ordered by capability (and usually by cost as well). In this setup, $\mathcal{G}_{\text{task}}$ expresses the structure of the problem by specifying which subtasks depend on others, while $(A, \leq_A)$ characterizes the solvers—showing which agent tiers are equipped to handle tasks of increasing difficulty and generality.

Assignment of tasks to agents can then be handled by a routing map $R : V_{\text{task}} \to A$. To ensure coherence, a natural constraint is monotonicity: if t is an ancestor of t' in $\mathcal{G}_{\text{task}}$, then $R(t) \geq_A R(t')$. This means that higher-level planning is given to more capable (typically more expensive) agents, while specific, lower-level execution is delegated to less capable tiers.