Memory has Many Faces: Simplicial Complexes as Agent Memory Layers

Luke Tandjung

***Abstract*—Current memory architectures for large language model agents employ vector embeddings and graph-based representations, with both approaches exhibiting significant performance gaps. While retrieval- augmented generation achieves strong results on benchmarks like 86% accuracy on LongMemEval, graph-based methods significantly underperform at 71%. This suggests that existing graph representations fail to capture essential memory structures. This paper proposes simplicial complexes as a unifying representation for agent memory that preserves higher-order relationships without the information loss inherent in pairwise graph projections. We introduce a database-backed simplex tree architecture that enables efficient storage and retrieval of multi-way interactions while supporting standard memory operations. Through a category-theoretic lens, we demonstrate that simplicial complexes occupy a privileged position in the hierarchy of knowledge representations, with structure-preserving functors to both graphs and vector spaces. Our approach enables the representation of contextual co-occurrence patterns (entities appearing together in conversations, or sessions) as geometric objects whose dimensional structure naturally encodes confidence and relationship strength.**

***Index Terms*—Agent Memory, Simplicial Complexes, Knowledge Graphs, RAG**

I. Introduction

Agent memory systems have converged on two paradigms: retrieving memories as vectors embeddings in semantic space, and knowledge graphs representing entity relationships explicitly. Despite recent advances, including hybrid architectures, performance gaps persist. On LongMemEval, RAG systems achieve 86% accuracy while graph-based methods reach only 71% [1].

This gap reflects a representational limitation. Vector embeddings collapse co-occurrence structure into continuous distances, losing discrete relationships. Knowledge graphs preserve structure but are constrained to pairwise interactions. When three or more entities co-occur meaningfully, such as multiple concepts in a conversation, or events in a session, pairwise graphs either discard this information [2] or attempt reconstruction through secondary inference [3]. Recent attempts at formalising this information loss include Wang and Kleinberg (2024), which proved the combinatorial impossibility of recovering higher-order structures from graph projections [4].

We propose simplicial complexes as a unifying representation that addresses these limitations. Simplicial complexes occupy a privileged position in the knowledge representation hierarchy, with structure-preserving functors to graphs and vectors while avoiding information loss from projection. The simplex tree data structure provides efficient database-backed implementation. This enables memory architectures preserving contextual co-occurrence patterns, yielding measurable improvements on multi-hop reasoning and temporal queries.

II. Motivation: A Category Theoretic Lens

Category. A category C consists of

- A collection of objects $a, b, c, \ldots$ denoted formally as the **class** $\mathrm{ob}(C)$.
- A collection of **morphisms** (arrows) $\hom_C(a, b)$ for each object pair a, b in $\mathrm{ob}(C)$. The expression $f : a \to b$ indicates that f is a morphism that maps a to b, as depicted in the commutative diagram below. The collection of $\hom_C(a, b)$ for all object pairs a, b in $\mathrm{ob}(C)$ is denoted as $\hom(C)$.

$$a \xrightarrow{f} b$$

- The composition binary operator $\circ$; that is, for any three objects a, b, c in $\mathrm{ob}(C)$,

$$\circ : \hom_C(a, b) \times \hom_C(b, c) \to \hom_C(a, b) \quad (1)$$

That is, given morphisms $f : a \to b$ and $g : b \to c$, the morphisms $g \circ f : a \to c$ exists, depicted below.

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[dr, "g \circ f"] & b \arrow[d, "g"] \\ & c \end{tikzcd}$$

Furthermore, the composition operation satisfies **associativity** and **identity** rules.

1) **Associativity**: For four objects a, b, c, d in $\mathrm{ob}(C)$ and morphisms $f : a \to b$, $g : b \to c$, $h : c \to d$, we have

$$(h \circ g) \circ f = h \circ (g \circ f) \quad (2)$$

2) **Identity**: For every object x in $\mathrm{ob}(C)$, there exists an identity morphism $\mathrm{id}_x : x \to x$ such that for every morphism $f : a \to b$, we have

$$\mathrm{id}_b \circ f = f = f \circ \mathrm{id}_a \quad (3)$$

Some examples of categories and their morphisms are sets and functions, groups and group homeomorphisms, vector spaces and linear maps, and topologies and continuous maps.

Functor. A functor $F : C \to D$ is a **structure-preserving** map between categories C and D. In particular,

- it assigns each object a in $\mathrm{ob}(C)$ an object $F(a)$ in $\mathrm{ob}(D)$.
- it assigns each morphism $f : a \to b$ in $\hom(C)$ a morphism $F(f) : F(a) \to F(b)$ in $\hom(D)$, such that $F(\mathrm{id}_x) = \mathrm{id}_{F(x)}$ and $F(f \circ g) = F(f) \circ F(g)$.

With that in mind, we can view representations as categories: vector embeddings as **Vect**, directed labelled graphs as **Vect**, and simplicial complexes as **SC**. Mapping from simplicial complexes to graphs happens via the **1-skeleton functor** F. Likewise, mapping from simplicial complexes to vector spaces happens via the **simplicial homology functor** H_k.

$$\begin{tikzcd} \mathrm{SC} \arrow[r, "F"] \arrow[dr, "H_k"] & \mathrm{Graph} \\ & \mathrm{Vect} \end{tikzcd}$$

However, as seen from the above diagram, no natural functor exists from graphs back to simplicial complexes. Given three mutually connected entities $\{a, b, c\}$, the graph doesn't indicate whether these entities co-occurred together (a filled triangle) or merely pairwise across contexts (an empty triangle) as seen in Fig. 1. This presents an implication for agent memory, When users discuss multiple concepts together, these co-occurrences carry semantic significance beyond pairwise projections. Graph-based systems either discard this at storage or attempt reconstruction at retrieval, both degrading performance.

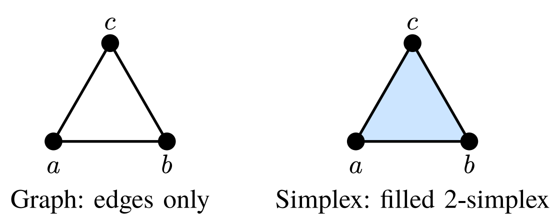

Fig. 1. A graph represents only pairwise edges, while a simplicial complex captures the 2-simplex $\{a, b, c\}$ as a filled face, encoding joint co-occurrence.

Simplicial complexes avoid this by representing higher-order co-occurences as first class objects. The functor hierarchy establishes representational power: complexes project to graphs and to embeddings, but systems at lower levels cannot recover richer structure. This points towards how memory systems should store information in the most structured form and project to coarser representations only when required.

III. Simplicial Complex for Knowledge

Simplicial Complex. A simplicial complex (Fig. 2) is a pair $R = (V, S)$ where

- V is the **vertex set** $V = \{v_1, v_2, ..., v_n\}$
- S is the **simplex/face set**. It is the set of non-empty subsets of V that satisfies the following properties by construction:
  1) $v \in V \Rightarrow \{v\} \in S$
  2) $s_1 \in S, s_2 \subseteq s_1 \Rightarrow s_2 \in S$. Here, s_2 is called the **face** of the simplex s_1. Furthermore, if $s_2 \subset s_1$, s_2 is called the **proper face** of the simplex s_1.

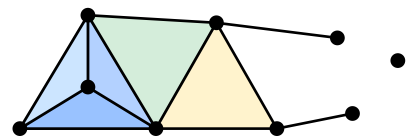

Fig. 2. A simplicial complex containing a 3-simplex (blue tetrahedron), 2-simplices (filled triangles), 1-simplices (edges), and 0-simplices (vertices). By downward closure, all faces of higher-dimensional simplices are automatically included.

The second property of **downward closure** for simplicial complexes distinguishes simplicial complexes from other possible candidates of knowledge representation like hypergraphs. If entities $\{v_1, v_2, v_3\}$ exists, then $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_1, v_3\}$, $\{v_1\}$, $\{v_2\}$, $\{v_3\}$ must exist. This captures semantic intuition: joint co-occurrence implies all sub-co-occurrences occurred. If three concepts were discussed together, each pair was discussed, and each individually. The way to describe such co-occurence is baked into the structure of simplicial complexes. If $s \in S$ has $k + 1$ elements, where $k \geq 0$, s is said to be a ***k*-simplex** of dimension k. A point is a 0-simplex, a line a 1-simplex, a filled triangle a 2-simplex, and a tetrahedron a 3-simplex (Fig. 3).

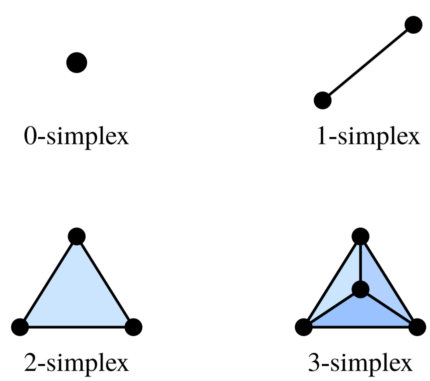

Fig. 3. Simplices of dimension 0 through 3: a vertex, edge, filled triangle, and tetrahedron.

For agent memory, this aligns with how knowledge emerges from contexts. A chat or search session referencing entities "neural networks", "backpropagation", and "gradient descent" produces a 2-simplex. By downward closure, this either creates edges for pairwise co-occurrences automatically, or implies the existence of co-occurences not yet uncovered, ensuring consistency. This idea is generalised in the structure of the simplicial complex through **facets**, which are the maximal dimension proper faces of a simplex. This involves taking the possible combinations of k points of a k-simplex; a 3-simplex $s = \{v_1, v_2, v_3, v_4\}$ will have the facets $\{v_1, v_2, v_3\}$, $\{v_2, v_3, v_4\}$, $\{v_1, v_3, v_4\}$, $\{v_1, v_2, v_4\}$. At the same time, the same simplex itself could be contained in another higher-dimensional simplices representing deeper structural pattern. Such a collection of higher-dimensional simplices are called **cofaces**; the cofaces of a 1-simplex $s = \{v_1, v_2\}$ could be something like $\{v_1, v_2, v_3\}$, $\{v_1, v_2, v_4\}$, $\{v_1, v_2, v_3, v_4\}$. More examples are shown in Fig. 4.

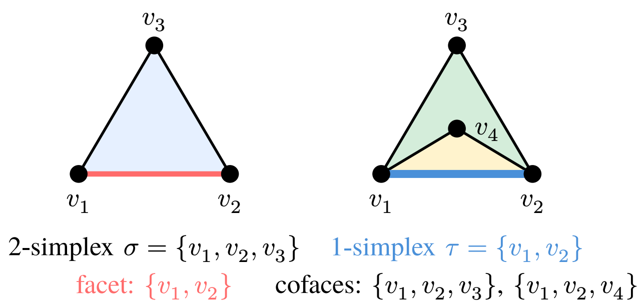

Fig. 4. Left: A facet of the 2-simplex $\{v_1, v_2, v_3\}$ is the edge $\{v_1, v_2\}$ (highlighted in red). Right: Cofaces of the 1-simplex $\{v_1, v_2\}$ (highlighted in blue) are the 2-simplices containing it.

These structures support natural memory operations. Coface lookup retrieves higher-dimensional simplices containing a query, identifying stronger contextual and narrative associations. Facet traversal descends to lower-dimensional faces, broadening the search when specificity must be relaxed.

When an agent retrieves knowledge relevant to a query, the retrieved simplices form a **subcomplex**, a simplicial complex $K' = (V', S')$ satisfying $V' \subseteq V$ and $S' \subseteq S$. However, behavioral data does not arrive in face-respecting order. A user may discuss entities $\{v_1, v_2, v_3\}$ together before ever discussing $\{v_1, v_2\}$ in isolation. The downward closure property defines a simplicial complex mathematically, but operational efficiency favours a different approach: we insert only directly observed simplices without materializing implied faces. A conversation yielding co-occurring entities $\{v_1, v_2, v_3\}$ produces a single 2-simplex insertion rather than the seven insertions required for full closure. At query time, faces that exist in the database represent independently confirmed co-occurrences, while faces computable from higher-dimensional simplices but absent from storage represent knowledge gaps—co-occurrences implied by context but never directly observed. At query time, this distinction becomes actionable.

Given a query-induced subcomplex formed by retrieving cofaces of semantically matched vertices, we enumerate the theoretical faces of each retrieved simplex and check their existence in storage. Faces that are combinatorially required but absent from the database represent knowledge gaps local to the query context: the system has evidence that entities co-occur in some higher-order relationship, but lacks direct confirmation of the supporting lower-order structure. The agent may then pose clarifying questions to confirm these missing relationships, or discount confidence in inferences that depend on unobserved faces.

This integrates naturally with **filtration**. A filtration of simplicial complex K is an ordering of K such that all prefixes are subcomplexes of K. That is, for two simplices σ, τ in K such that $\sigma \subset \tau$, σ appears before τ in the ordering. For example, consider building a filled triangle $\{v_1, v_2, v_3\}$. A valid filtration orders the simplices as:

$$\begin{aligned} &\{v_1\} \to \{v_2\} \to \{v_3\} \to \{v_1, v_2\} \to \\ &\{v_2, v_3\} \to \{v_1, v_3\} \to \{v_1, v_2, v_3\} \end{aligned} \tag{4}$$

Each prefix forms a valid subcomplex: vertices appear before edges, and edges before the filled face (Fig. 5).

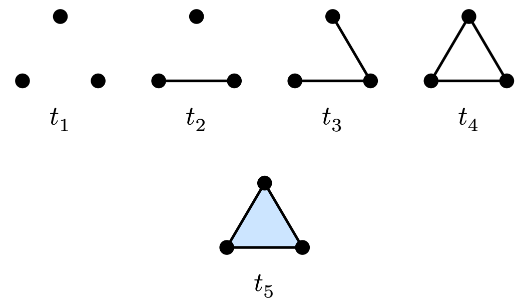

Fig. 5. A filtration building a 2-simplex. At t_1, only vertices exist. Edges are added at t_2-t_4. The filled 2-simplex appears at t_5, only after all its faces exist.

Rather than treating filtration as a global property, we compute it over the query-induced subcomplex. Closure-induced faces can be flagged for inference. The agent may then pose clarifying questions or bias search results to confirm missing relationships, or discount confidence in inferences that depend on closure-induced faces. The topology of the query-induced subcomplex thus serves not only as a retrieval mechanism but as an inference guide, directing attention toward gaps most relevant to the immediate task.

IV. Simplex Trees

The simplex tree, introduced by Boissonnat and Maria in 2014, provides an efficient data structure for representing abstract simplicial complexes of arbitrary dimension[5]. The simplex tree reconciles the need to explicitly store all faces of the complex with the desire for compact representation and efficient operations, making it particularly well-suited for database-backed memory systems.

For the simplicial complex $K = (V, S)$ of dimension k (that is, the dimension of the largest simplicial complex in K), we label each vertice $v_i \in V$ a letter $l_i \in \{1, ..., |V|\}$ from the alphabet $1, ..., |V|$, where $1 \leq l_1 < ... < l_{|V|} \leq |V|$. Then, the simplex $s = \{v_1, ..., v_i\}$ can be represented as the word $[s] = [l_1, ..., l_j]$. This allows us to express every k-simplex $s \in S$ as a word of $(k+1)$ length with labels of ascending order. We start with an empty **trie**, and construct the tree as such (Fig. 6):

- Words are inserted from the root or node to the leaf of the tree, with the first letter as the root or node and the last letter as the leaf.
- If inserting a word $[s] = [l_1, ...l_j]$, and the longest prefix word already in the tree is $\{l_1, ..., l_i\}$, of which $i < j$, we append the rest of the word $[l_{i+1}, ..., l_j]$ to the l_i node.

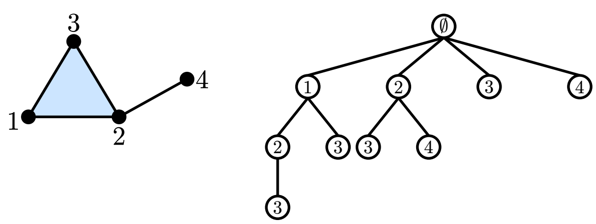

Fig. 6. A simplicial complex K with 2-simplex $\{1, 2, 3\}$ and edge $\{2, 4\}$ (left) and its simplex tree representation (right). Each path from root to node encodes a simplex: e.g., $\emptyset \to 1 \to 2 \to 3$ represents $\{1, 2, 3\}$.

For an in-memory implementation, there are additional requirements:

1) A red-black tree or hash table is usually used for the trie structure, with the top nodes using an array.
2) Each sibling node has a pointer to its parent node.
3) For all nodes in the same tree depth of the same letter label l_i, they are connected in a circular linked list.

The pseudocode implementation is given below.

References

[1] Emergence.ai, "SOTA on LongMemEval with RAG." [Online]. Available: https://www.emergence.ai/blog/sota-on-longmemeval-with-rag

[2] V. Salnikov, D. Cassese, R. Lambiotte, and N. Jones, "Co-occurrence simplicial complexes in mathematics: identifying the holes of knowledge," *Applied Network Science*, vol. 3, no. 1, 2018, doi: 10.1007/s41109-018-0074-3.

[3] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial Closure and higher-order link prediction," *arXiv preprint arXiv:1802.06916*, 2018, doi: 10.48550/arxiv.1802.06916.

[4] Y. Wang and J. Kleinberg, "From Graphs to Hypergraphs: Hypergraph Projection and its Remediation," *arXiv preprint arXiv:2401.08519*, 2024, doi: 10.48550/arxiv.2401.08519.

[5] J.-D. Boissonnat and C. Maria, "The Simplex Tree: an Efficient Data Structure for General Simplicial Complexes," *arXiv preprint arXiv:2001.02581*, 2020, doi: 10.48550/arxiv.2001.02581.