The Imagination Machine XI: Graph-Theoretic Realizations of Compression and Extension

Mark Tracy
Boston University
mrktracy@bu.edu

March 2026

Abstract

The Imagination Machine series develops a formal architecture for embedded epistemic systems based on recursive cycles of compression and extension. The present paper develops this architecture in two directions. First, we show that graph theory provides a natural concrete realization: graph quotients implement compression, graph completion implements extension, and compression–extension dynamics on graphs induce simplicial dynamics on their clique complexes, connecting relational networks to the simplicial architecture of the series. Second, we extend this realization to a computational architecture for unsupervised learning in interactive text environments. An agent embedded in such an environment maintains a knowledge graph whose vertices are entity embeddings and whose edges are learned relation weights. The agent compresses this graph by clustering entities in embedding space, extends it by prompting a language model to complete partial relational configurations, and acts by generating text conditioned on the compressed graph. The supervision signal is entirely internal: the agent predicts its own next graph state and updates in response to prediction error. We characterize the fixed points of this dynamics as epistemically closed world models in the sense of The Imagination Machine I, identify conditions under which the dynamics stabilize, and connect the resulting architecture to graph neural networks, topological data analysis, and knowledge graph reasoning. The language model in this architecture is not the imagination machine — it is the extension operator. The imagination machine is the full compression–extension–action–observation loop.

Contents

1 Introduction

An agent wakes up with nothing. No labels, no teacher, no prior knowledge of the domain it has been placed in. Observations arrive as text. The agent has no way to step outside its observational surface to check whether what it believes about the world is correct. It has access only to what passes through that surface — and only to the consequences of its own actions on what passes through next.

This is the setting of The Imagination Machine I, stated concretely. The agent is embedded in the sphere. The sphere's interior is all it has.

What can such an agent learn? The answer the series gives is: the relational invariants of its environment — the structure that persists across observations, that survives compression, that keeps being confirmed by the consequences of action. Not the world as it is, but the world as it appears from inside a particular observational surface, compressed to the resolution that proves predictively useful.

The present paper develops this answer in two stages.

The first stage is mathematical. We show that graph theory provides the natural concrete realization of the compression–extension architecture. Graphs encode relational structure directly. Graph quotients implement compression by collapsing entities that are indistinguishable under the agent's current world model. Graph completion implements extension by reconstructing relational structure consistent with preserved invariants. And compression–extension dynamics on graphs induce genuine simplicial dynamics on their clique complexes — face maps for compression, simplicial completion operations for extension — connecting this concrete realization to the abstract simplicial structure identified in The Imagination Machine X.

The second stage is computational. We develop a concrete architecture in which a language model serves as the extension operator in an unsupervised learning loop. The agent maintains a knowledge graph whose vertices are entity embeddings and whose edges are learned relation weights. At each step it compresses the graph by clustering entities in embedding space, extends the compressed graph by prompting the language model to complete partial relational configurations, acts by generating text conditioned on the compressed graph, observes the environment's response, and updates in response to the difference between its predicted graph state and the actual graph state that resulted.

The supervision signal is entirely internal. The agent predicts its own next representational state — not the raw observation, but the update to its own knowledge graph that the observation will induce. The target of prediction is the world model itself. The fixed point of this loop is a world model that accurately predicts its own updates: a model that has internalized the structure of the environment deeply enough that new observations no longer surprise it at the representational level. This is epistemic closure from the inside of the sphere.

A clarification that the architecture requires: the language model in this system is not the imagination machine. It is the extension operator — one component of the loop. The imagination machine is the full compression–extension–action–observation cycle, of which the language model's generative capacity is one part. An LLM without compression, without action, without the feedback of prediction error against subsequent observation, is not an imagination machine. It is a completion engine. What makes the system an imagination machine is the loop.

Section 2 through Section 6 develop the mathematical foundations. Section 7 presents the computational architecture. Section 8 gives the full algorithm. Section 9 addresses stabilization and convergence. Section 10 connects the architecture to related work.

2 Graphs as Relational Structures

Definition 1 (Graph)**.** *A graph is a pair $G = (V, E)$ where V is a set of vertices and $E \subseteq \binom{V}{2}$ is a set of edges.*

Vertices represent entities and edges represent binary relations between entities. Graphs constitute the minimal relational structure: they encode which pairs of entities stand in a given relation without imposing additional algebraic or metric constraints.

Definition 2 (Graph Morphism)**.** *A graph morphism $\phi : G \to G'$ is a map $\phi : V \to V'$ such that $(u, v) \in E$ implies $(\phi(u), \phi(v)) \in E'$.*

Graph morphisms are the structure-preserving maps between relational structures, forming the morphisms of the category **Graph**.

Remark 1. *The connection to The Imagination Machine I is direct. Observational profiles $\gamma \in \Gamma$ encode the relational structure the agent can access. A graph $G = (V, E)$ is a relational structure in the same sense: V indexes the entities present in the agent's observational field and E records which pairs stand in the observed relation. The inference map $F : \Gamma \to W$ is, in the graph-theoretic realization, a compression of the observed relational graph into a world model.*

3 Compression as Graph Quotient

Definition 3 (Graph Quotient)**.** *Let $G = (V, E)$ be a graph and let $\sim$ be an equivalence relation on V. The quotient graph $G/\!\sim$ has vertex set $V/\!\sim$ and edge set*

$$E/\!\sim = \big\{\, ([u], [v]) : \exists\, u' \in [u],\, v' \in [v] \text{ with } (u', v') \in E,\ [u] \neq [v] \,\big\}.$$

The quotient map $q : G \to G/\!\sim$ sends each vertex to its equivalence class.

Proposition 1. *The quotient map $q : G \to G/\!\sim$ is a graph morphism.*

Proof. By definition of $E/\!\sim$, an edge $([u], [v]) \in E/\!\sim$ exists if and only if there exist $u' \in [u]$ and $v' \in [v]$ with $(u', v') \in E$. Thus $q(u') = [u]$, $q(v') = [v]$, and $(q(u'), q(v')) \in E/\!\sim$, confirming that q is a graph morphism. □

Remark 2. *The equivalence relation $\sim$ encodes the relational invariants the compressing agent chooses to preserve. Vertices equivalent under $\sim$ are indistinguishable from the perspective of those invariants. This is precisely the role of the equivalence relation $d \sim_w d'$ induced by a world model w in The Imagination Machine I: two observations are equivalent when the world model assigns them to the same representational class. Graph quotient is the graph-theoretic instance of that operation. Graph clustering, coarsening, and community detection are all instances of graph compression in this sense.*

4 Extension as Graph Completion

Definition 4 (Graph Completion)**.** *Let $G = (V, E)$ be a graph representing a partial relational configuration. A completion of G with respect to a constraint set $\mathcal{C}$ is a graph $G' = (V', E')$ such that $V \subseteq V'$, $E \subseteq E'$, and every added edge or vertex is consistent with $\mathcal{C}$.*

The constraint set $\mathcal{C}$ plays the role of the world model: it encodes the relational regularities that extension must respect. Extension is not arbitrary addition of structure but constrained generation consistent with preserved invariants.

Remark 3. *The implication map $g : W \to \Gamma$ of The Imagination Machine I is, in the graph-theoretic realization, exactly this operation: given a world model, generate the predicted relational structure. Link prediction, motif inference, and generative graph models are all instances of graph completion.*

5 The Clique Complex

Definition 5 (Clique)**.** *A clique in $G = (V, E)$ is a subset $C \subseteq V$ such that every pair of vertices in C is connected by an edge.*

Definition 6 (Clique Complex)**.** *The clique complex $X(G)$ of a graph $G = (V, E)$ is the simplicial complex whose simplices are the cliques of G:*

$$X(G) = \{\, C \subseteq V : C \text{ is a clique in } G \,\}.$$

Proposition 2. *$X(G)$ is a simplicial complex.*

Proof. If σ is a clique and $\tau \subseteq \sigma$, then every pair of vertices in τ is also a pair in σ, hence connected. Thus τ is a clique and $\tau \in X(G)$. □

Proposition 3. *Let G and G' be graphs with $G \subseteq G'$, meaning $V(G) \subseteq V(G')$ and $E(G) \subseteq E(G')$. Then the induced map*

$$X(G) \hookrightarrow X(G')$$

is an inclusion of simplicial complexes.

Proof. Every simplex of $X(G)$ is a clique in G. Since $G \subseteq G'$, every edge present between vertices of that clique in G is also present in G'. Therefore every clique of G is also a clique of G', so every simplex of $X(G)$ is a simplex of $X(G')$. Hence $X(G)$ is a simplicial subcomplex of $X(G')$, and the induced map is an inclusion. □

Remark 4. *This proposition shows that graph extension lifts monotonically to the simplicial level: adding relational structure to a graph enlarges its clique complex by simplicial inclusion. This monotone lifting is the graph-theoretic counterpart to the extension direction of the compression–extension cycle: just as the implication map $g : W \to \Gamma$ generates richer observational profiles from compressed world models, graph completion generates richer simplicial structure from compressed relational graphs.*

The face maps of $X(G)$ are given by vertex deletion: for a k-simplex $\sigma = [v_0, \ldots, v_k]$, the i-th face map is $\partial_i \sigma = [v_0, \ldots, \hat{v}_i, \ldots, v_k]$.

6 Compression–Extension Dynamics

Definition 7 (Compression–Extension Update)**.** *A compression–extension step is a pair of operations*

$$G_t \xrightarrow{C_t} H_t \xrightarrow{E_t} G_{t+1}$$

where C_t is a graph compression (quotient by $\sim_t$) and E_t is a graph completion (extension consistent with $\mathcal{C}_t$).

Lemma 1. *Let $q : G \to G/\sim$ be a quotient map merging two adjacent vertices u and v. Then the induced map $X(q) : X(G) \to X(G/\sim)$ acts as a simplicial face map on every simplex containing both u and v.*

Proof. Let $\sigma = [v_0, \ldots, v_k] \in X(G)$ contain both $v_i = u$ and $v_j = v$. Under q, both map to $[u]$. The image $q(\sigma)$ is the clique $[q(v_0), \ldots, \widehat{q(v_j)}, \ldots, q(v_k)]$, which is the face obtained by removing v_j — exactly the action of ∂_j on σ. For simplices not containing both u and v, q restricts to a bijection on the vertex set and clique structure is preserved by Proposition 1. □

Lemma 2. *Let $e : G \to G'$ be a completion adding a single edge (u, v) where u and v belong to a common clique $\sigma \in X(G)$. Then $X(e)$ extends the simplicial structure by adding new simplices corresponding to newly formed cliques.*

Proof. Adding (u, v) may unify cliques containing u and v respectively into a single larger clique in G'. The resulting clique corresponds to a higher-dimensional simplex in $X(G')$. Thus the map $X(e)$ extends the simplicial complex by including new simplices generated by the enlarged clique structure. □

Theorem 1. *Let (G_t) be a sequence of graphs generated by compression–extension updates. Then the induced sequence of clique complexes $(X(G_t))$ evolves through simplicial operations: compression steps induce face maps and extension steps induce simplicial completion operations that enlarge the clique complex by inclusion when new cliques are formed.*

Proof. By Lemma 1, each compression step induces face maps on the clique complex. A general quotient decomposes into elementary vertex merges, each inducing a face map; their composition is a simplicial map. By Lemma 2, each extension step enlarges the clique complex by simplicial inclusion through the addition of new simplices corresponding to newly formed cliques. A general completion decomposes into elementary edge additions, each inducing such a simplicial completion; their composition is a simplicial map. The composite $X(G_t) \to X(H_t) \to X(G_{t+1})$ is therefore a composition of face maps followed by simplicial completion maps, which is a simplicial map. □

Corollary 1. *If the extension operator satisfies the horn-filling condition — every partial relational configuration admitting a consistent completion receives one — then $(X(G_t))$ satisfies the Kan condition.*

Remark 5. *The connection between Proposition 3 and Corollary 1 is direct. A horn $\Lambda^k_i \hookrightarrow \Delta^k$ in the clique complex is a partial clique configuration with one face missing. The horn-filling condition requires that whenever such a partial configuration is consistent with the constraint set $\mathcal{C}$, the extension operator completes it by adding the missing simplex. By Proposition 3, this completion corresponds to a graph extension $G \subseteq G'$ that adds the missing edges, and the induced inclusion $X(G) \hookrightarrow X(G')$ supplies the missing simplex. The Kan condition is therefore the requirement that the extension operator is complete with respect to the constraint set: no consistent horn goes unfilled.*

7 A Computational Architecture for Unsupervised Learning

7.1 The Setting

The agent is embedded in an interactive text environment. Observations arrive as strings. Actions are strings. The environment has its own relational structure — entities, relations, causal dependencies — which the agent cannot observe directly. It observes only the textual surface of that structure.

The agent begins with nothing: no entities, no relations, no prior model of the domain. It must construct its world model entirely from the inside, using only the observations it receives and the consequences of its own actions. This is the condition of maximal epistemic aloneness: the agent has no external teacher, no ground truth signal, no vantage point outside the sphere.

7.2 The Knowledge Graph as World Model

The agent's world model is a knowledge graph with two components:

- $V \in \mathbb{R}^{n \times d}$: a matrix of entity embeddings, one row per entity, each row a dense vector in the language model's representation space.
- $E \in [0,1]^{n \times n \times r}$: a sparse tensor of relation weights, where $E[i,j,k]$ is the agent's current confidence that relation k holds between entity i and entity j.

The graph is not a static symbolic database. It is a living structure that grows, compresses, and refines with each observation. Entities are not discrete symbols but continuous vectors; the "symbol" is the embedding. Relations are not binary but graded by confidence.

7.3 The Language Model as Extension Operator

A pretrained language model serves as the extension operator $g : W \to \Gamma$. It is called in two modes:

- **Extraction**: given a raw text observation o_t, extract entity–relation–entity triples. This is the grounding operation that converts the unstructured observational surface into symbolic relational content.
- **Completion**: given the compressed graph H_t serialized as structured text, predict missing relations and implied entities. This is the extension operation that fills horns in $X(H_t)$ — completing partial relational configurations consistent with the constraint set.

The language model does not maintain the knowledge graph. The knowledge graph is maintained by the agent. The language model is a tool the agent uses to update and extend its graph — not the agent itself.

7.4 Predicting the Next Graph State

The critical feature of the architecture is what it predicts. The agent does not predict the next raw observation o_{t+1}. It predicts the next graph state G_{t+1} — the update to its own world model that the next observation will induce.

The target of prediction is the world model itself. The supervision signal is the difference between the predicted graph $G_{t+1}^{\text{predicted}}$ and the actual graph G_{t+1}^{actual} that results from observing o_{t+1}:

$$\mathcal{L}_t = \text{diff}(G_{t+1}^{\text{predicted}}, G_{t+1}^{\text{actual}})$$

where diff counts the symmetric difference between predicted and actual edge sets. This loss is entirely internal: it requires no external label, no ground truth, no oracle. The environment provides the next observation; the agent's own representational machinery converts that observation into a graph update; the difference between predicted and actual update is the error signal.

The agent is modeling its own modeling process. It is predicting its own next predicted update.

8 Algorithm

We present the full algorithm. All types are defined in Section 7.

Types

$V : \mathbb{R}^{n \times d}$	entity embedding matrix
$E : [0,1]^{n \times n \times r}$	relation weight tensor
$G = (V, E)$	knowledge graph
$o_t \in \text{Text}$	observation
$a_t \in \text{Text}$	action

Subroutines

```
UPDATE_GRAPH(G, o_t):
  triples ← LLM.extract(o_t)
    // prompt: "extract (entity, relation, entity)
    //          triples from: [o_t]"

  for each (e1, rel, e2) in triples:

    v1 ← LLM.encode(e1)
    s  ← cosine(v1, V)
    if max(s) > sim_thresh:
      i    ← argmax(s)
      V[i] ← mean(V[i], v1)     // update existing entity
    else:
      i    ← len(V)
      V    ← append(V, v1)      // add new entity

    v2 ← LLM.encode(e2)
    s  ← cosine(v2, V)
    if max(s) > sim_thresh:
      j    ← argmax(s)
      V[j] ← mean(V[j], v2)
    else:
      j    ← len(V)
      V    ← append(V, v2)

    r        ← relation_index(rel)
    E[i,j,r] ← 1.0              // observed: full confidence

  return (V, E)

COMPRESS(G, sim_thresh):
  S        ← cosine(V, V)       // pairwise similarity matrix
  clusters ← union_find(S, sim_thresh)
                                // merge i,j if S[i,j] > sim_thresh
  V_new ← []
  idx   ← {}                    // old index → new index
  for each cluster c:
    V_new    ← append(V_new, mean(V[i] for i in c))
```

```
    for i in c:
      idx[i] ← len(V_new) - 1

  E_new ← zeros(len(clusters), len(clusters), r)
  for each (i,j,k) with E[i,j,k] > 0:
    E_new[idx[i], idx[j], k] ← max(
      E_new[idx[i], idx[j], k],
      E[i,j,k]
    )

  return (V_new, E_new)          // H_t = G_t / ~_t

EXTEND(H_t, LLM):
  prompt     ← serialize(H_t)
    // "known entities: [...]
    //  known relations: [...]
    //  predict missing or implied relations:"

  candidates  ← LLM.complete(prompt)
  G_predicted ← copy(H_t)

  for each (e1, rel, e2) in candidates:
    i ← resolve(e1, V, sim_thresh)
    j ← resolve(e2, V, sim_thresh)
    k ← relation_index(rel)
    G_predicted.E[i,j,k] ← 0.5  // predicted: half confidence

  return G_predicted

DIFF(G_pred, G_actual):
  // count edges in symmetric difference:
  // edges predicted but not observed +
  // edges observed but not predicted
  return |{(i,j,k) : G_pred.E[i,j,k] > 0}
        △ {(i,j,k) : G_actual.E[i,j,k] > 0}|
```

Main Loop

```
INITIALIZE:
  G          ← ([], [])         // empty graph
  sim_thresh ← 0.8              // compression threshold
  η          ← 0.01             // threshold learning rate
  t          ← 0

for t = 0, 1, ..., T:

  // OBSERVE
  o_t ← environment.observe()

  // UPDATE
  G_t ← UPDATE_GRAPH(G, o_t)

  // COMPRESS
  H_t ← COMPRESS(G_t, sim_thresh)
```

```
// EXTEND
G_predicted ← EXTEND(H_t, LLM)

// ACT
a_t ← LLM.act(serialize(H_t), task)
environment.act(a_t)

// OBSERVE OUTCOME
o_{t+1}  ← environment.observe()
G_actual ← UPDATE_GRAPH(H_t, o_{t+1})

// COMPUTE ERROR
error ← DIFF(G_predicted, G_actual)

// REWARD (unsupervised)
r_process     ← -error
r_compression ← -len(H_t.V) / max(len(G_t.V), 1)
reward        ← β * r_process + γ * r_compression

// REFINE
sim_thresh ← sim_thresh + η * sign(error - error_prev)
  // high error → raise threshold → coarser compression
  // low error  → lower threshold → finer compression

update_policy(reward)

G          ← G_actual
error_prev ← error
```

Convergence Condition

The algorithm converges to an RL closure (w^*, π^*) when:

error $\to 0$	graph accurately predicts its own updates
$\lvert H_t.V \rvert \to$ stable	compression has stabilized
sim_thresh $\to$ stable	granularity has stabilized

9 Stabilization and Convergence

The architecture does not guarantee convergence. Whether the dynamics stabilize depends on three conditions.

Environmental stability. The environment must have stable relational structure. If the world keeps changing its rules — if the relations that hold at time t are systematically different from those that hold at time $t+1$ — the knowledge graph cannot converge. The agent will keep revising its model in response to observations without ever reaching a fixed point. This is not a failure of the architecture; it is the correct behavior of an embedded agent in a genuinely nonstationary environment. The quasi-periodic setting of The Imagination Machine III is the minimal environment in which convergence is guaranteed, because the environment has exact invariants — the frequency ratios — that the agent can recover.

Compression aggressiveness. If the similarity threshold `sim_thresh` is too low, the graph accumulates entities without merging them. The vertex set grows without bound and the compression step fails to reduce representational complexity. The adaptive threshold update in the main loop addresses this: persistent high prediction error raises the threshold, forcing coarser compression and preventing graph bloat.

LLM consistency. If the language model produces inconsistent completions — predicting different relations for the same compressed graph on different calls — prediction error will not decrease even if the world model is otherwise accurate. This is the Kan condition stated as a practical requirement on the extension operator: the language model must be able to fill every consistent horn, and must do so consistently. In terms of Corollary 1, the horn-filling condition is a testable property of the language model that determines whether the induced clique complex sequence satisfies the Kan condition.

The garbage filter. The architecture is not garbage-in-garbage-out in a naive sense. Edges that do not predict future observations generate high prediction error and are penalized through the reward signal. Stable structure — relations that keep being confirmed by action–observation cycles — survives compression. The compression–extension cycle is hostile to relational content that does not earn its place. Whether the filter is strong enough depends on the ratio of signal to noise in the environment and the expressiveness of the compression operator. This is an empirical question that the toy implementation of The Imagination Machine III is designed to address in the minimal quasi-periodic setting.

10 Implications

Graph neural networks. Message passing in a GNN is a compression operation: it collapses the local relational neighborhood of each vertex into a compressed feature vector. Theorem 1 implies that GNN dynamics induce simplicial dynamics on the clique complexes of their input graphs. GNNs are implicitly performing face map operations on simplicial complexes, and their expressive power is bounded by the simplicial structure they can detect.

Topological data analysis. A compression–extension orbit (G_t) defines a filtration of clique complexes $(X(G_t))$. The persistent homology of this filtration captures the relational invariants that survive across compression–extension cycles — precisely the fixed points of the closure operator $T = F \circ g$ in the graph-theoretic realization.

Knowledge graph reasoning. Reasoning over knowledge graphs involves both compression (identifying equivalent entities) and extension (inferring missing relations). The present paper establishes that this reasoning process has a simplicial interpretation, connecting knowledge graph operations to the homotopy-theoretic properties of the Kan complex identified in The Imagination Machine X.

Large language models. The architecture clarifies the role of language models in embedded epistemic systems. A language model is a powerful extension operator: it can complete partial relational configurations, infer missing entities, and generate text consistent with a compressed world model. But it is not, by itself, an embedded epistemic agent. It lacks compression, action, and the feedback loop that drives prediction error to zero. Embedding a language model in the

compression–extension–action–observation loop of the present architecture is what promotes it from completion engine to component of an imagination machine.

11 Conclusion

The Imagination Machine architecture describes recursive cycles of compression and extension governing representation and reasoning in embedded epistemic systems.

The present paper has developed this architecture in two directions. Mathematically, graph quotients implement compression and graph completion implements extension, and the resulting dynamics induce simplicial face maps and simplicial completion operations on associated clique complexes. Computationally, a language model serving as extension operator — embedded in a loop with a knowledge graph, a compression step, an action channel, and an internal prediction error signal — realizes the architecture as a concrete unsupervised learning system.

The agent in this system learns from maximal epistemic aloneness. It has no teacher, no labels, no external ground truth. It has only its observations, the consequences of its actions, and the difference between what it predicted its world model would become and what it actually became. The structure that crystallizes from this process — the entities that survive compression, the relations that keep being confirmed, the graph that stabilizes — is the agent's answer to the question of what its environment is.

That answer may be wrong. Convergence is not guaranteed. The filter may be too weak, the language model too inconsistent, the environment too nonstationary. These are empirical questions. But the conditions under which the answer is right are precisely the conditions under which an embedded agent can know anything at all: a world stable enough to have invariants, a compression aggressive enough to find them, and an extension consistent enough to test them.

The imagination machine does not know the world from outside. It constructs the world from within the only closure available to it.