# PUTMAN Model

## A Core Architecture for Contextual Meaning Reconstruction

Stephen A. Putman

### Abstract

PUTMAN is a core architecture for modeling interpretation shift in recursive symbolic systems. It represents meaning as weighted relational structure that is selectively activated under contextual constraints and reconstructed into an interpretation under limited information. Overlap between relational structures (literal or structural) induces recombination during reconstruction, accounting for ambiguity, semantic drift, conflict, and emergent reinterpretation under repetition. The model also includes modulation mechanisms—affect, recursion depth, and rigidity—that control activation and suppression dynamics and thereby shape reconstruction stability. Core objects and operators are defined once in Section 2.0 (Canonical Definitions), and the remainder of the paper provides minimal examples and implementation targets that follow directly from those definitions. A minimal evaluation protocol and reference pseudocode are provided to make interpretation shift measurable and the architecture testable.

### 1. Introduction

Many semantic systems implicitly assume that symbols map to stable meanings. In practice, interpretation is context-dependent and changes under repetition, affect, and memory. The same surface token can reliably produce different reconstructions across speakers, across time, or across states of pressure, even when wording remains unchanged. These shifts are not rare edge cases; they are a routine consequence of reconstruction from partial structure under constraint.

Interpretation shift is the change in reconstructed interpretation I produced from the same stored representation M under varying context C, recursion depth d, affect modulation E, or rigidity ρ; formally, it is a measurable difference between $I_t$ and $I_{t+1}$ under a chosen distance metric. Related notions appear in adjacent literature as semantic drift (linguistics/NLP) and concept drift (machine learning), but PUTMAN focuses on within-system reconstruction under contextual constraints.

PUTMAN addresses interpretation shift by representing meaning as relational structure that is selectively activated under contextual constraints and reconstructed into an interpretation. The goal is not an ontological claim about what meaning "is," but a modeling layer for systems that must track mismatch, recombination, and instability in recursive communication. Context does not contribute "additional meaning" as content; it governs which relations become active and which remain latent.

Section 2 defines the core objects and operators used throughout the paper. Section 3 specifies the modulators that govern reconstruction stability. Section 4 provides minimal examples mapping directly to the architecture. Sections 5–7 outline implementation targets, limits, and evaluation directions. Section 8 provides a minimal protocol for measuring interpretation shift, and Appendix A provides reference pseudocode for a concrete instantiation.

## 2. Core Model: Meaning as Relational Structure

PUTMAN models meaning as relational structure rather than a linear definition. A token or symbol is not treated as an atomic label, but as a structured bundle of relations that can be activated, suppressed, and recombined under constraint. The core claim is architectural: the same stored structure can yield multiple valid reconstructions depending on context, and overlap between structures is a primary driver of interpretation shift, ambiguity, and emergent reinterpretation.

### 2.0 Canonical Definitions, Types, and One Concrete Instantiation

This paper treats the following objects and operators as modeling commitments (representational assumptions), not ontological claims.

A stored meaning representation is $M = (V, E, w, t)$, where $V$ is the node set, $E$ is the edge set, $w : E \to \mathbb{R}$ assigns edge weights, and $t : E \to \mathcal{T}$ assigns relation types. Nodes and edges are representational anchors used for reconstruction under constraint.

$C$ denotes a context package, modeled as a set of constraints and priors relevant to interpretation (task, discourse history, speaker model, tone, and memory state). For implementability, we treat context as producing parameter values that modulate activation and pruning (e.g., bias terms or thresholds) rather than as additional semantic content.

Activation. Produces an activated subgraph S of (V, E). Activation may be deterministic or stochastic; to make the architecture testable, we specify one default stochastic instantiation: each edge $e \in E$ is included with probability $p(e \mid C, E) = \sigma(\alpha \cdot w'(e) + b\_C(t(e)))$, where $\sigma$ is sigmoid, $\alpha$ is a scaling constant, $b\_C(\cdot)$ is a context-dependent bias by relation type, and $w'(e)$ is the affect-modulated weight defined below. The activated subgraph S is formed by sampling edges (and their incident nodes) under this policy.

Reconstruction. Maps activated structure and context into an interpretation object I. For implementability, we instantiate I as a ranked list of candidate reconstructions produced by constrained search over S under C (e.g., beam search), where each candidate's score aggregates satisfied constraints and weighted evidence from S. This keeps reconstruction operational: it is a procedure that returns ranked candidates rather than an informal "reconstruction idea."

Admissibility. The admissible interpretation space $\mathcal{A}(C, \rho)$ is defined as the set of candidate interpretations whose score meets a rigidity-dependent threshold: $\mathcal{A}(C, \rho) = \{ I : \mathrm{score}(I; S, C) \geq \tau(\rho) \}$. This makes admissibility testable.

Rigidity. $\rho \in [0, 1]$ parameterizes pruning strength and branching control. Operationally, we set an admissibility score threshold $\tau(\rho) = \tau\_min + \rho(\tau\_max - \tau\_min)$ and optionally tie beam width to rigidity via $k(\rho) = k\_max - \lfloor \rho(k\_max - k\_min) \rfloor$. Higher $\rho$ therefore reduces branching and increases pruning pressure in a specified way.

Overlap. For two stored structures $M_1$ and $M_2$, let $S_1 = A\_C(M_1)$ and $S_2 = A\_C(M_2)$ denote their activated subgraphs under the current context. We operationalize overlap with at least two measures: (a) literal overlap via Jaccard similarity on node or edge sets of $S_1$ and $S_2$, and (b) structural overlap via cosine similarity of subgraph embeddings (e.g., spectral or message-passing embeddings) computed on $S_1$ and $S_2$. These measures are used to predict when recombination is likely under partial activation.

Affect and recursion depth. E denotes an affect modulation signal (scalar or vector) and d denotes recursion depth (number of reconstruction passes). For a minimal instantiation, affect is implemented as a bias on edge weights $w'(e) = w(e) + \beta \cdot E \cdot s(t(e))$, where $s(t)$ selects which relation types are affected. Recursion depth is implemented as repeated passes that update weights after reconstruction: $w_{i+1}(e) = w_i(e) + \eta \cdot \mathrm{presence}(e, I_i) - \lambda \cdot \mathrm{decay}(e)$, where presence(·) indicates whether e supports the selected reconstruction and decay(·) prevents unbounded growth.

**2.0.1 Notation (Quick Reference)**

| Symbol | Meaning |
|---|---|
| M | stored meaning representation (weighted, typed relational graph) |
| C | context package (constraints/priors modulating activation/pruning) |
| S | activated subgraph of M under C |
| I | reconstructed interpretation output (selected or ranked candidates) |
| ρ (rho) | rigidity (pruning / branching control) |
| $\mathcal{A}$(C, ρ) | admissible interpretation set under context C at rigidity ρ |
| E | affect modulation signal |
| d | recursion depth (number of reconstruction passes) |
| $\mathcal{T}$ | finite relation-type vocabulary used by **t : E →** $\mathcal{T}$ (e.g., association, contrast, role, affordance, affect) |

**2.1 Representation (Meaning Graph)**

PUTMAN represents meaning as a weighted, typed relational graph whose nodes denote symbols, concepts, or representational anchors, and whose edges capture relations such as association, contrast, role, affordance, and affect linkage. The weight function w allows the same structure to support different reconstructions under different contexts. The type function t supports modeling differences between relations that may otherwise be conflated (e.g., a role relation versus an affect relation), which becomes important when overlap induces recombination.

In this framing, “compressed” does not imply a literal compression algorithm. It denotes that many downstream interpretations can be generated from the same stored relational structure via selective activation and constrained reconstruction, rather than requiring separate stored definitions for each use-case.

**2.2 Contextual Activation**

PUTMAN treats context C as an activation and constraint package. It governs which parts of M become available for reconstruction and how strongly they compete. Operationally, context modulates activation biases and thresholds, shaping which relations become active versus latent under C.

This separation is deliberate: the stored structure M can remain stable while activation changes across situations, enabling context-sensitive reconstruction without requiring that the underlying representation be replaced each time the same token appears.

### 2.3 Constrained Reconstruction

PUTMAN treats interpretation as the output of constrained reconstruction from partial structure under context. Reconstruction uses the activated subgraph S and constraints C to produce candidate interpretations and select among them. Because the activated structure is partial and constraints vary, reconstruction is not guaranteed to be unique. Multiple candidate interpretations may be admissible, and different agents may converge on different candidates under different contexts even when surface text is identical.

This framing explains mismatch as a structural phenomenon: coherent reconstructions can disagree because they were produced from different activated subgraphs or under different constraint packages, not because one side "failed to retrieve the definition."

### 2.4 Overlap & Recombination (Mechanism of Interpretation Shift)

Interpretation shift becomes likely when partially overlapping structures compete during reconstruction. Overlap may be literal (shared nodes or edges) or structural (near-isomorphic substructures under a similarity mapping). When context activates an overlapping region, reconstruction can incorporate relations supported by both structures, producing recombination. Depending on constraints and pruning, recombination can yield ambiguity, gradual semantic drift, conflict, or emergent reinterpretation.

Under PUTMAN, such outcomes are modeled consequences of reconstruction from partially overlapping relational structure under shifting constraints.

### 2.5 PUTMAN Pipeline Summary

PUTMAN models interpretation using an invariant pipeline:

relational encoding → contextual activation → constrained reconstruction

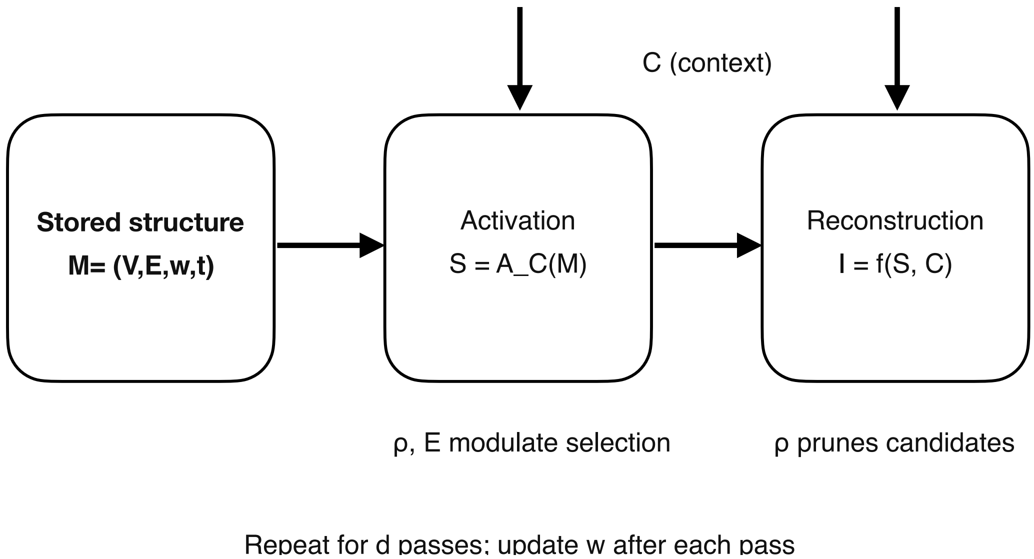


**Figure 1.** PUTMAN operational pipeline. Stored structure M is activated under context C to produce S, and reconstruction produces interpretation I. Rigidity ρ and affect modulation E shape selection and pruning; recursion depth d repeats the cycle via weight updates.

## 3. Modifiers: Affect, Recursion Depth, and Rigidity

In recursive symbolic systems, interpretation is not computed once. Outputs influence later activations, weights shift under affect modulation, and constraints tighten or loosen over time. PUTMAN models affect, recursion depth, and rigidity as modulators that shape activation, suppression, and reconstruction outcomes, determining whether reconstruction stabilizes or whether interpretation shift accelerates.

### 3.1 Affect as a Modulation Signal

PUTMAN does not require a complete theory of affect. It models affect operationally as a modulation signal E that biases activation and suppression dynamics. Affect can be implemented as a bias term on relation weights, as a modifier on activation probability for specific relation types, or as a modifier on suppression thresholds during reconstruction.

Operationally, affect changes which relations dominate under constraint. Under the same surface token and broadly similar context, different affect states can produce different activated subgraphs and different reconstructions, even when both reconstructions remain internally coherent.

### 3.2 Recursion Depth and Reconstruction Pressure

Recursion depth d denotes the number of repeated reconstruction passes in which prior outputs influence later activation and reconstruction. Higher recursion depth increases reconstruction pressure in two ways. First, reconstructed interpretations become part of the effective context for later passes, shifting activation policies. Second, repeated selection and suppression reinforce some relations and weaken others over time, producing stable or unstable reconstruction regimes.

Under low recursion depth, reconstruction may remain close to baseline activation. Under higher depth, small differences in context, affect, or overlap can compound into significant interpretation shift, especially when multiple partially overlapping structures compete.

### 3.3 Rigidity as Branching Control

Rigidity ρ parameterizes pruning strength and branching control across activation and reconstruction. Higher ρ implies stronger suppression of alternative relations, reduced branching in activated structure, and more winner-take-all reconstruction behavior.

Rigidity shapes the admissible interpretation space $\mathcal{A}(C, \rho)$ by shrinking the effective hypothesis space available to reconstruction. Communication failures and meaning clashes can be modeled as insufficient overlap between admissible spaces across agents. Importantly, failure is not always caused by absence of shared structure. Many high-friction cases occur when overlap exists in principle, but is operationally inaccessible due to asymmetric pruning.

## 4. Minimal Examples

Each example below demonstrates a single PUTMAN mechanism without expanding into broad cultural interpretation. The goal is to show how relational structure, activation, reconstruction, and pruning interact to produce interpretation shift.

### 4.1 Wolf (Time-Indexed Semantic Drift Under Recursion)

What it demonstrates. A stable token can be associated with different time-indexed relational structures, producing semantic drift as reconstruction repeats under changing contexts and repeated exposure.

Setup. Consider the token “wolf” under two time indices, represented as $M_{t_1}$ and $M_{t_2}$. The node set may overlap substantially while relation weights and types shift. At $t_1$, high-weight relations may emphasize threat, predation, livestock loss, and antagonistic narrative roles. At $t_2$, higher weights may emphasize ecosystem role, conservation framing, managed risk, and protected status, while threat relations may remain but change weight or type.

Mechanism. Under stable context conditions, repeated reconstruction reinforces the relations that remain consistently activated, producing a stable reconstruction regime. Under different stable contexts, a different subset becomes dominant. Over time, the reconstructed interpretation associated with the token changes as the dominant activated relations change across contexts and across repeated passes.

Why it matters. This example demonstrates how long-horizon semantic drift can be modeled as a trajectory produced by repeated constrained reconstruction from time-indexed relational structure, rather than as a change in a single definition.

### 4.2 Snake (Cross-Domain Partial Overlap and Structural Similarity)

What it demonstrates. Two agents or domains can reconstruct meaning from partially overlapping structures even when label identity is weak. Structural similarity can support alignment without literal node matching, while overlap can also induce recombination under partial activation.

Setup. Let agent A and agent B carry meaning graphs M_A and M_B shaped by different domains. Both may include relations around danger, concealment, venom, avoidance, and vigilance, while diverging elsewhere. Overlap can occur as shared subgraph or as near-isomorphic relational shape where roles align despite label differences.

Mechanism. Under context constraints, each agent activates a subset of its stored structure and reconstructs an interpretation. Alignment can occur when activated substructures are structurally similar, even if surface labels differ. Divergence occurs when reconstruction draws on different high-weight relations outside the overlapping region, or when overlap induces recombination that changes implication structure under constraint.

Why it matters. This example shows why overlap alignment is not a simple word-match problem. PUTMAN provides a principled way to model structural similarity, partial overlap, and recombination as drivers of interpretation shift in cross-domain communication.

### 4.3 Jargon (Rigidity Collapsing Admissible Space After Agreement)

What it demonstrates. Communication failure can occur after agreement when rigidity collapses branching width and suppresses bridge relations, reducing overlap between admissible spaces toward zero.

Setup. Suppose two participants share a goal anchor g and agree at the goal level. Divergence occurs at the first branching step: the set of admissible solution branches differs due to constraints and pruning policies. Jargon is treated here as an activation-and-pruning regime: within a group it may efficiently activate a shared subgraph, but under high rigidity it can also collapse branching width and exclude adjacent alternative branches.

Mechanism. As rigidity increases, admissible interpretations under the group's constraints shrink. Bridge relations that would otherwise support mapping become operationally inaccessible. The result is that two agents can share a goal anchor while losing enough overlap in admissible spaces to reconstruct compatible interpretations.

Why it matters. This reframes disagreement as a structural phenomenon driven by branching control, not necessarily by vocabulary or bad faith. It also gives PUTMAN an operator-level handle for intervention: widening admissible space, negotiating constraints explicitly, or restoring access to bridge relations.

## 5. Applications

PUTMAN is intended as a foundational architecture for systems that must track interpretation shift under context, affect modulation, and repetition. The following applications are presented as implementation targets rather than claims of domain authority.

### 5.1 Dialogue and Assistive Agents

In multi-turn interaction, agents face symbol reuse where surface text remains stable while implied interpretation changes. PUTMAN supports tracking these changes as shifts in activated structure and reconstructed interpretation across turns, enabling drift detection and mismatch alerts without requiring static label mappings.

### 5.2 Narrative and NPC Systems

Narrative systems and NPC agents require stable representation with context-sensitive interpretation. PUTMAN supports this by separating stored relational structure from activation and reconstruction, allowing affect and recursion modulation to shape behavior coherently without brittle rule explosion.

### 5.3 Communication Mediation

PUTMAN's rigidity parameter provides an operator-level handle for failures driven by pruning. Systems can detect collapsing overlap in admissible spaces and respond by widening branching, negotiating constraints, or restoring access to bridge relations. The aim is not forced convergence, but restoration of sufficient shared structural space for productive reconstruction.

## 6. Limits & Scope

This paper is intentionally narrow. It specifies a core architecture for modeling interpretation shift via relational representation, contextual activation, and constrained reconstruction, and for explaining ambiguity and drift as modeled outcomes of reconstruction under constraint.

PUTMAN does not claim to be a complete theory of cognition, a comprehensive semantic ontology, or a clinical framework. It does not attempt to fully model affect; it treats affect as an operational modulation signal. It does not require that meaning graphs be “true” in an external sense; it models reconstruction behavior given representational assumptions. Applications are presented as implementation targets and design motivations, not as evidence of domain-level efficacy.

## 7. Conclusion & Future Work

PUTMAN formalizes interpretation shift in recursive symbolic systems using a small set of architectural commitments: relational representation, contextual activation, and constrained reconstruction, with explicit modulators for affect, recursion depth, and rigidity. Interpretation shift is treated as measurable change in reconstructed interpretation under varying constraints and pruning policies, rather than as failure of static definitions.

Future work should focus on operationalizing and evaluating the architecture. Key directions include methods for estimating activation and overlap in real time (including structural similarity measures that do not depend on label identity), distance metrics for measuring interpretation shift across recursion depth, explicit pruning policies mapping rigidity to algorithmic mechanisms, and minimal reference implementations that demonstrate the PUTMAN pipeline in toy graphs and applied settings. The objective is to keep the model grounded in observable reconstruction behavior: how interpretations change when stored relational structure is selectively activated and reconstructed under constraint.

## 8. Evaluation: Measuring Interpretation Shift (Minimal Protocol)

Interpretation shift is measured as change in reconstructed interpretation across steps under controlled changes to C, E, ρ, or recursion depth d. We treat this as a distance between interpretation outputs $I_t$ and $I_{t+1}$ under an explicit metric.

Distance metrics. Two minimal metrics are sufficient to make interpretation shift testable: (1) graph-edit distance between reconstructed subgraphs associated with interpretations, and (2) embedding distance between interpretation representations (e.g., cosine distance on fixed-length interpretation embeddings). Either metric enables quantitative tracking of shift; using both guards against artifacts from a single representation.

Synthetic benchmark. To demonstrate the mechanics without domain claims, a synthetic generator can create meaning graphs with typed relations, controllable overlap, and controllable ambiguity. A simple benchmark varies (a) overlap between two graphs, (b) rigidity ρ, and (c) recursion depth d, and records how often reconstruction switches between candidate interpretations and how distances accumulate over time.

Protocol sketch. For each condition: generate M, sample a sequence of contexts $C_t$ (including perturbations), run activation → reconstruction for t = 1..T, update weights across recursion depth passes, and compute $\Delta(I_t, I_{t+1})$. Report mean and variance of shift magnitude, switch rate between top candidates, and overlap statistics.

This protocol is sufficient to validate that interpretation shift is measurable, controlled by the proposed modulators, and sensitive to overlap and rigidity as claimed.

## Appendix A: Minimal Reference Pseudocode (Activation + Pruning + Reconstruction)

```
Inputs: M, C, ρ, E, d    # ρ = rho (rigidity)
Params: α, β, η, λ, τ_min, τ_max, k_min, k_max

τ = τ_min + ρ*(τ_max − τ_min)
k = k_max − floor(ρ*(k_max − k_min))

for depth i in 1..d:

  # Affect-modulated weights
  for each edge e in E:
    w'(e) = w(e) + β*E*s(t(e))

  # Stochastic activation
  S = empty subgraph
  for each edge e in E:
    q = sigmoid(α*w'(e) + b_C(t(e)))
    if rand() < q:
      include e (and incident nodes) in S

  # Constrained reconstruction (beam search)
  # BeamSearch returns candidate interpretations I
  candidates = BeamSearch(S, C, beam_width=k)
  I_i = argmax_{I in candidates} score(I; S, C)

  # Weight update (reinforce used edges, decay)
  for each edge e in E:
    w(e) = w(e) + η*presence(e, I_i) − λ*decay(e)

Output: final interpretation I_d; shift metrics vs prior step
```