Mimir: A Foundational Architecture for a Goal-driven Cognitive Core

Your Name

October 5, 2025

Abstract

This document outlines the highest level of the Mimir system's cognitive architecture: a goal-driven, predictive core that enables true agency. This active and sophisticated mechanism does not merely maintain a stateful "thread of thought"; it actively models possible futures, evaluates them against its intrinsic objectives, and uses the error between its predictions and reality to learn. This forms a continuous, self-improving loop where the agent learns to steer itself towards states that it predicts will be more optimal. We detail the architecture of the core conscious state, the Actor-Critic learning loop that drives behavior, the self-regulatory mechanisms for managing cognitive resources, and the protocols that allow this architecture to extend from a single mind to a multi-agent collective intelligence.

1 Introduction

Modern AI systems have achieved remarkable results in specialized domains, but often lack the characteristics of true agency: goal-directed behavior, self-improvement, and the ability to reason about and plan for the future. This paper introduces the Mimir cognitive core, an architecture designed to provide these capabilities.

The core is built around a predictive-evaluative loop. Instead of simply reacting to stimuli, the agent actively models potential future states, evaluates them against a learned value function representing its core objectives, and selects actions that are predicted to lead to more optimal states. The feedback loop between prediction and reality provides a continuous learning signal, allowing the agent to improve its policies and its model of the world over time.

This paper specifies the architecture of this cognitive core. We detail the structure of the conscious state vector, the Actor-Critic learning mechanism, the role of the Policy Actor in decision-making and self-regulation, and finally, the protocols that enable communication and cooperation within a multi-agent ecosystem.

2 The Two-Tiered Temporal Engine

The perceptual pipeline (detailed in the companion paper) produces a rich, multi-scale representation of the world at a single moment in time. To enable reasoning about dynamics, sequences, and narratives, the system employs a two-tiered temporal architecture that models changes at vastly different timescales.

2.1 Tier 1: The Temporal Dynamics Engine (TDE)

The TDE is the short-term "physics engine" of the system, responsible for modeling smooth, moment-to-moment transitions.

2.1.1 Architecture and Input

The TDE is implemented as an RWKV-style WKV recurrence (or a similar state-space model such as a Transformer). At each cognitive cycle t, it receives the complete state vector S_t, which is the concatenation of:

1. The dense, invariant multimodal concept vector $C_{M,inv}(t)$ (the output of the $Proteus - M \rightarrow Codec - M$ block).
2. The collection of unimodal equivariant state vectors $\{\Phi_{V,eq}(t), \Phi_{T,eq}(t), \Phi_{A,eq}(t), \ldots\}$.

The complete state is thus: $S_t = [C_{M,inv}(t) \oplus \Phi_{V,eq}(t) \oplus \Phi_{T,eq}(t) \oplus \cdots]$.

2.1.2 Function

In analysis mode, the TDE processes the stream $\{S_t\}$ sequentially, learning to predict S_{t+1} given the history $(S_t, S_{t-1}, \ldots)$. In synthesis mode, the TDE generates a coherent sequence of future states $\{S_{t+1}, S_{t+2}, \ldots\}$, optionally conditioned on a high-level script from the narrative engine.

2.2 Tier 2: The Proteus Temporal Pyramid

The Proteus Temporal Pyramid is the long-term "narrative engine," designed to discover abstract, recurring patterns in how concepts evolve over time.

2.2.1 Architecture and Input

The pyramid is a hierarchical, bottom-up structure built from chained Proteus instances. It operates not on the high-frequency stream of complete state vectors $\{S_t\}$, but on time-blocked sequences of only the abstract, invariant concept vectors $\{C_{M,inv}(t)\}$.

At the first level of the pyramid, the stream of concept vectors is segmented into discrete time blocks (e.g., one-minute blocks):

$$Block_1 = \{C_{M,inv}(t) \mid t \in T_1\}, \quad Block_2 = \{C_{M,inv}(t) \mid t \in T_2\}, \ldots$$

Each block is the input dataset for a corresponding Proteus run ($Run(T_1)$, $Run(T_2)$, etc.).

2.2.2 Warm-Start Chain and Dynamics Vectors

The first block, $Run(T_1)$, is a cold-start run executing the full Proteus Stage 1/2 pipeline. Subsequent blocks use a warm-start mechanism: $Run(T_2)$ begins with the converged hierarchy from $Run(T_1)$, skips Stage 1, and proceeds directly to Stage 2 refinement. This allows the system to compute a `dynamics_vector` by comparing the final hierarchy of $Run(T_2)$ to its initial, warm-started state.

2.2.3 Bottom-Up Temporal Tree

Higher-order, longer-term patterns are discovered by recursively running new Proteus instances on the outputs of lower-level runs. The outputs of $Run(T_1)$ and $Run(T_2)$ are merged (with statistical pruning to remove obsolete clusters) and fed into a higher-level $Run(T_1^1)$. This process continues, forming a deep, hierarchical tree that models the narrative structure of events at increasing timescales (minutes, hours, days, etc.).

2.3 Bidirectional Flow: Analysis and Synthesis

The two tiers work in concert to enable both bottom-up understanding and top-down generation.

Analysis (Bottom-Up).

1. The Perceptual Pipeline produces the stream $\{S_t\}$.
2. The TDE processes this stream continuously to model immediate dynamics.
3. The abstract portion $\{C_{M,inv}(t)\}$ is buffered and periodically dispatched to the Proteus Temporal Pyramid for long-term structural analysis.

Synthesis (Top-Down).

1. A high-level generative goal is defined as a target region on the Proteus Temporal Pyramid's manifold (e.g., "generate a scene about a character asking a question").
2. The pyramid samples a trajectory, producing a sequence of target abstract concepts: $\{C^*_{M,inv}(t+1), C^*_{M,inv}(t+2), \ldots\}$. This is the "script."
3. The script is passed as a conditioning signal to the TDE.
4. The TDE generates a sequence of full state vectors $\{S_t\}$ that are both locally coherent and goal-directed (matching the script).
5. The sequence $\{S_t\}$ is dispatched to the unimodal students for rendering into pixels, audio, and text.

This two-tiered architecture elegantly separates the "physics" of the world (the TDE) from the "story" of the world (the Temporal Pyramid), creating a robust foundation for temporal reasoning at all scales.

3 The Cognitive Core Architecture

The foundation of the cognitive core is a single, high-dimensional **Conscious State Vector (v_{Ct})**, representing the system's complete, integrated awareness at a moment in time. This vector is updated in a discrete cognitive cycle.

3.1 State Integration and the Cognitive Cycle

At each timestep, a "Cognitive Bootstrapping" process intelligently selects the most relevant information from five distinct streams: current perception, recalled perceptual memories, the prior conscious state, recalled cognitive memories, and a privileged symbolic command channel.

3.1.1 The Five Input Streams

The raw cognitive input $v_{cognitive\_raw}$ is formed by concatenating summaries from five sources:

1. **Current Perception ($v_{perceptual\_summary}$):** The complete state vector S_t from the Temporal Dynamics Engine, containing both the current multimodal concept $C_{M,inv}(t)$ and all equivariant state vectors $\{\Phi_{eq}(t)\}$.
2. **Recalled Perceptual Memories:** Entity vectors retrieved by the Entity Management System (EMS) in response to the current perception.
3. **Prior Conscious State ($v_{C,t-1}$):** The conscious state from the previous cognitive cycle, providing internal context.
4. **Recalled Cognitive Memories:** Previously-experienced thought patterns retrieved by the EMS in response to the prior state.
5. **Symbolic Command ($v_{symbolic\_command}$):** Input from the Symbolic Execution Environment, such as user instructions or internal queries.

3.1.2 Abstraction via Proteus-Conscious

The high-dimensional, concatenated $v_{cognitive\_raw}$ is not used directly. Instead, a dedicated Proteus instance, **Proteus-Conscious**, is trained on the stream of these raw vectors. It learns the manifold of "states of mind," discovering recurring patterns in the interplay between perception, memory, and internal context. Its output is a sparse, variable-dimension fuzzy membership vector v_I representing the system's complete situation at time t.

3.1.3 Densification and State Update

The sparse v_I is passed through a dedicated **Codec Autoencoder** to produce a dense, fixed-size vector $v_{I,dense}$. This codec is trained to map the sparse conceptual address to a rich, operational embedding suitable for integration with the recurrent state. The conscious state is then updated via a multi-rate leaky integrator: $v_{C,t} = Update(v_{C,t-1}, v_{I,dense})$.

3.2 The Policy Actor and Self-Regulation

The updated conscious state v_{Ct} is fed into the **Policy Actor**, the primary decision-making engine. This component, a specialized Component-Level ANFIS-GNN, outputs a policy delta, Δ_π, containing the system's high-level intention for its next action.

3.2.1 The Component-Level ANFIS-GNN Architecture

The Policy Actor is designed to reason about the relationships between all components of the conscious state in an isotropic manner, avoiding the sequential bias of recurrent architectures.

Graph Construction. The graph is built at maximal granularity: **every dimension of the conscious state vector $v_{C,t}$ becomes a node**. If $v_{C,t} \in R^{2048}$, the graph has 2048 nodes. Each node's initial feature is its scalar value from $v_{C,t}$, augmented with a learnable type embedding indicating its origin (e.g., perceptual concept, memory, self-regulatory parameter).

Learned Sparse Connectivity. Edges are not fully connected. The system uses a **Graph Attention Network (GAT)** mechanism where each node learns attention scores for its potential neighbors, dynamically creating a task-specific sparse graph at each cycle. An inductive bias is provided: nodes from the same source stream (e.g., all perceptual features) have denser initial connectivity, while cross-stream edges are sparser.

GNN Reasoning and ANFIS Decoder. Through several message-passing rounds (3–6 layers), each node updates its state by aggregating information from its attended neighbors. The final node states are aggregated and fed into a **Multi-Output ANFIS Decoder**. This decoder has a shared fuzzy rule base (AND/OR layers) that interprets the aggregated GNN state, followed by separate, independent output heads for each dimension of the policy delta Δ_π.

Self-Regulatory Actions. Crucially, Δ_π includes proposed updates not only for external actions but also for the agent's own internal cognitive parameters, which are themselves part of the conscious state. These parameters include:

- Global learning rate (α)

- Attentional workspace capacity (k, m)
- Predictive planning depth $(k_{cognitive})$

This allows the system to learn an optimal policy for managing its own cognitive resources, such as "thinking further ahead" for complex tasks or reducing its learning rate when its model of the world is stable.

3.3 The Symbolic Execution Environment (SEE)

The symbolic command channel provides an interface to a sandboxed SEE. This environment is the system's "logical body," allowing it to act upon the world programmatically. It executes commands from the Policy Actor, manages I/O for generative tasks, and can even host persistent, event-driven triggers, allowing the agent to build its own fast-acting reflexes.

4 The Predictive-Evaluative Learning Loop

The mechanism that drives all high-level learning and goal-seeking behavior is an **Actor-Critic** architecture, adapted from reinforcement learning. The system learns both how to evaluate states (the Critic) and how to choose actions that lead to better states (the Actor).

4.1 The Critic: A Learned Value Function

The Critic is an ANFIS module, $V(s)$, whose job is to learn the expected objective score (or "value") for any given conscious state s. After the system takes an action a_t from state s_t and observes the actual reward r_t at the new state s_{t+1}, the Critic is trained to minimize the **Temporal Difference (TD) Error**:

$$Error = r_t + V(s_{t+1}) - V(s_t)$$

This error signal represents how much better or worse the outcome was than expected, and is used to make the Value Function a more accurate predictor of future rewards.

4.2 The Actor: An Improved Policy Function

The Actor is the Policy Actor module, $\pi(a|s)$, which generates the action a_t from state s_t. The core of the learning process is using the TD Error (now termed the "Advantage", A_t) to improve the policy. The system reinforces actions that lead to positive advantage and suppresses actions that lead to negative advantage. The update rule for the policy's weights (θ) is a policy gradient ascent, scaled by this advantage:

$$\theta_{t+1} = \theta_t + \alpha_t \cdot A_t \nabla_\theta \log \pi_\theta(\Delta_t)$$

This directly trains the agent to choose actions that it predicts will maximize its objective score over time, forming a continuous, self-improving loop.

5 Self-Reflection and Cognitive Memory

To enable introspection and the ability to learn from past experience, the system employs a dedicated mechanism for analyzing its own thought patterns.

5.1 Proteus-Reflexive: Learning the Manifold of Thought

A high-level Proteus instance, **Proteus-Reflexive**, is trained on the historical sequence of the system's own conscious state vectors $\{v_{C,t}\}$. This instance learns the manifold of "states of mind," discovering recurring cognitive patterns (e.g., problem-solving, observation, planning).

5.2 Cognitive Schemas and Efficient Recall

The clusters discovered by Proteus-Reflexive represent specific, recurring thought patterns or "cognitive schemas." These are successful sequences of mental states that have led to good outcomes in the past. When the Policy Actor recognizes a situation similar to one it has encountered before, it can choose to recall and execute a pre-recorded schema from the Entity Management System (EMS), bypassing the expensive multi-step prediction process. This provides a powerful mechanism for routine cognitive tasks to be handled efficiently, while preserving the capacity for deliberate, multi-step reasoning when novel situations arise.

5.3 Entity Management System (EMS)

The EMS acts as a fast, reflexive k-NN retrieval system for both perceptual and cognitive memories. When a new instance (perceptual or cognitive) is identified, the EMS automatically performs a k-NN search against its persistent index. Retrieved memories are then included in the five-stream input to Proteus-Conscious, allowing past experiences to inform current decisions.

6 Multi-Agent Systems and Collective Intelligence

The Mimir architecture can be extended beyond a single agent to support complex multi-agent systems. We define two primary design patterns for such systems, distinguished by their approach to the credit assignment problem.

6.1 Design Pattern 1: Monolithic Agents in a Decentralized Ecosystem

This is the default extension, where multiple, independent Mimir agents interact in a shared, untrusted environment. Each agent is a self-interested individual operating on its own Actor-Critic learning loop. In this "anarchy" of agents,

interactions are governed by game theory and security protocols, as there is no central authority to compel or reward cooperation.

6.2 Design Pattern 2: The Swarm Entity

This pattern creates a single, cohesive consciousness that inhabits a distributed physical body (e.g., a swarm of drones). The individual drones are subordinate agents—the "limbs" of a central "swarm consciousness." This central mind uses a **Group-Relative Policy Optimization (GRPO)** style mechanism to solve the credit assignment problem. It evaluates the performance of each drone relative to its peers and sends back targeted learning signals to improve the efficiency and coordination of the entire swarm. This is not a society of minds, but a single, distributed organism.

7 Common Vocabulary Reconciliation

For multiple Mimir agents to communicate effectively, they must first establish a shared understanding of the concepts they are discussing. The Symbolic Execution Environment (SEE) manages a **Common Vocabulary Reconciliation** protocol to achieve this alignment.

The protocol is comprehensive, aiming to align concepts across all shared modes of experience:

- **Perceptual Concepts:** Agents align their understanding of sensory data by exchanging and cross-analyzing benchmark datasets using their respective 'Proteus-M' models.
- **Cognitive Concepts:** Agents align their models of thought itself by exchanging common cognitive schemas (successful sequences of past actions) or their 'Proteus-Reflexive' models of state transitions.
- **Episodic & Entity Memory:** Agents exchange identifiers for key public entities or events (e.g., "The 2024 Olympics") to create a shared frame of reference.

Once a common vocabulary is established, agents can engage in hyper-efficient communication. Instead of serializing a concept into language, an agent can transmit the single, rich, fuzzy membership vector that represents that concept. The receiving agent treats this vector as a first-class input to its own cognitive loop, allowing for the high-bandwidth exchange of abstract ideas between trusted agents.

8 Conclusion

We have presented the Mimir cognitive core, an architecture for agentic, goal-driven intelligence. By combining a predictive-evaluative loop based on an

Actor-Critic model with a self-regulating policy, the system learns to actively steer itself toward more optimal states over time. The architecture's core components—the unified conscious state, the symbolic execution environment, and the protocols for multi-agent communication—provide a robust foundation for a single agent to reason, act, and learn. Furthermore, the specified design patterns for multi-agent systems and vocabulary reconciliation extend these capabilities from a single mind to a cooperative collective intelligence. This work provides a blueprint for moving beyond narrow AI to more general, adaptive, and purposeful intelligent systems.

References