An Idea of a Neural Web(Larkos)

Witold Warchoł

January 25, 2025

Last updated: February 15, 2026

1 Introduction

This document outlines the mathematical framework of the Lattice-Associated Recursive Knowledge Organization System (neural web architecture). The complete system integrates dynamic neuron specialization with memory, emotion, and predictive systems to create an adaptive artificial intelligence framework.

2 Design of Larkos

Larkos is designed with several key principles in mind:

1. **Hierarchical Memory System**: The network uses a hierarchical memory system with short-term, medium-term, and long-term memory clusters. This design allows the network to prioritize and retain important information while discarding less relevant data.

2. **Dynamic Adaptation**: The network dynamically adjusts its parameters based on performance and stability. This ensures that the network can adapt to new data and changing conditions, improving its robustness and accuracy.

3. **Non-linear Activation**: The use of the tanh function introduces non-linearity into the neuron outputs, enabling the network to model complex relationships in the data.

4. **Memory Consolidation and Decay**: Memories are consolidated based on their importance and decay over time. This mimics the natural process of memory retention and forgetting, ensuring that the network remains efficient and focused on relevant information.

5. **Performance Optimization**: The network continuously monitors its performance and optimizes its parameters to achieve the best results. This includes adjusting learning rates, batch sizes, and other hyperparameters.

6. **Pattern Matching and Similarity**: The network uses cosine similarity to compare memory vectors and identify similar patterns. This allows the network to recognize and recall relevant information from its memory.

7. **Predictive Coding**: The network employs predictive coding to anticipate future states and improve its predictive accuracy.

8. **Advanced Neuron Management**: The network dynamically manages its neurons, adding or removing them based on their performance to maintain optimal network efficiency.

9. **Meta-Controller**: The network uses a meta-controller to manage the importance and learning efficiency of different regions, enhancing overall learning and adaptation.

10. **Context Management**: The network organizes and updates context nodes to maintain a coherent global context, adapting to environmental factors and constraints.

11. **Self-Reflection**: The network performs self-reflection to analyze its performance and coherence, detect potential confabulation, and regenerate responses.

12. **Self-Identity**: The network manages its identity components based on behavioral patterns and experiences, providing insights into its consistency and confidence.

13. **Knowledge Filtering**: The network categorizes inputs and records problem instances, analyzing category statistics to update category importance and confidence.

14. **Emotional Processing**: The network incorporates an emotional system that influences decision-making and learning based on emotional states like love and hate, providing affective modulation of cognitive processes.

15. **Imagination Capability**: The network can simulate hypothetical scenarios and outcomes before taking action, enabling more sophisticated planning and decision-making.

16. **Social Intelligence**: The network models social interactions and maintains representations of other agents, allowing for more natural and effective social behavior.

17. **Specialization**: The network performs specialization to optimize performance and adaptability.

3 Neuron Specialization System

3.1 Initialization

The specialization system is initialized with:

$$S = (\mathcal{N}, \theta_t, \mathcal{D}) \tag{1}$$

where:

- $\mathcal{N}$ = Set of specialized neurons
- θ_t = Specialization threshold
- $\mathcal{D}$ = Type distribution vector

3.2 Specialization Types

The system recognizes eight specialization types:

1. **Pattern Detector (SPEC_PATTERN_DETECTOR)**: Identifies specific input patterns
2. **Feature Extractor (SPEC_FEATURE_EXTRACTOR)**: Extracts consistent features
3. **Temporal Processor (SPEC_TEMPORAL_PROCESSOR)**: Processes time-dependent patterns
4. **Context Integrator (SPEC_CONTEXT_INTEGRATOR)**: Combines multiple information sources
5. **Decision Maker (SPEC_DECISION_MAKER)**: Drives output decisions
6. **Memory Encoder (SPEC_MEMORY_ENCODER)**: Stabilizes important information
7. **Emotional Processor (SPEC_EMOTIONAL_PROCESSOR)**: Handles affective responses
8. **Prediction Generator (SPEC_PREDICTION_GENERATOR)**: Anticipates future states

3.3 Detection Algorithm

For each neuron n_i, specialization scores are computed:

$$s_j(n_i) = f_j(\text{output}, \text{state}, \text{connections}, \text{temporal\_behavior}) \tag{2}$$

where f_j is the scoring function for specialization type j.

3.3.1 Scoring Functions

$$\text{Pattern Score} = \begin{cases} 0.8 & \text{if } o_i > 0.7 \wedge c_{\text{input}} > 0.6 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\text{Temporal Score} = 0.7 + 0.2(1 - \sigma_t) \tag{4}$$

$$\text{Memory Score} = 0.5 + 0.3o_i + 0.1\mathbb{I}(k_i > 4) \tag{5}$$

where:

- o_i = neuron output
- c_{input} = input correlation
- σ_t = temporal variance
- k_i = number of connections
- $\mathbb{I}$ = indicator function

3.4 Specialization Application

For each specialized neuron $n_i \in \mathcal{N}$ with type t_i and boost factor β_i:

$$\text{Update}(n_i) = \begin{cases} \text{state} \leftarrow \text{state} \cdot (1 + 0.2\beta_i) & t_i = \text{Pattern} \\ \text{state} \leftarrow \text{state} + 0.05\beta_i(\text{state} - \mu_i) & t_i = \text{Temporal} \\ \text{weights} \leftarrow \text{weights} \cdot (1 + 0.1\beta_i) & t_i = \text{Context} \\ \text{output} \leftarrow \text{contrast\_enhance}(\text{output}, \beta_i) & t_i = \text{Decision} \end{cases} \tag{6}$$

3.5 Importance Adaptation

The importance factor β_i updates as:

$$\beta_i^{(t+1)} = \text{clip}\left(\beta_i^{(t)} + \eta(\alpha S_a + (1-\alpha)S_p - 0.5), 0.1, 2.0\right) \tag{7}$$

where:

- S_a = activity score (type-specific)
- S_p = network performance score
- α = blending factor (0.6)
- η = learning rate (0.1)

3.6 System Evaluation

Overall effectiveness:

$$E = 0.4P + 0.3\bar{\beta} + 0.3\left(-\sum_{j=1}^{8} p_j \log p_j\right) \tag{8}$$

where P is network performance and p_j is the proportion of type j.

4 Memory Vector Computation

The memory vector is computed by combining neuron states, neuron outputs, and input tensor values. The formula for computing the memory vector is:

$$\text{memory\_vector}[i] = \begin{cases} \text{neurons}[i].\text{state} & \text{if } i < \text{MAX\_NEURONS} \\ \text{neurons}[i - \text{MAX\_NEURONS}].\text{output} & \text{if MAX\_NEURONS} \leq i < 2 \times \text{MAX\_NEURONS} \\ \text{input\_tensor}[i - 2 \times \text{MAX\_NEURONS}] & \text{if } 2 \times \text{MAX\_NEURONS} \leq i < \text{MEMORY\_VECTOR\_SIZ} \\ 0 & \text{otherwise} \end{cases}$$

The memory vector is a concatenation of neuron states, neuron outputs, and input tensor values. If the vector size exceeds the available data, the remaining elements are set to zero. This ensures that the memory vector has a fixed size, which is necessary for consistent processing.

5 Importance Score Calculation

The importance score for a memory vector is calculated as the average of the absolute values of its elements:

$$\text{importance} = \frac{1}{\text{MEMORY\_VECTOR\_SIZE}} \sum_{i=0}^{\text{MEMORY\_VECTOR\_SIZE}-1} |\text{memory\_vector}[i]|$$

The importance score measures the overall significance of a memory vector. Higher absolute values in the vector indicate greater importance. This score is used to determine which memories should be retained, consolidated, or forgotten.

6 Memory Decay

Memories decay over time based on a decay factor. The decay formula is:

$$\text{importance}_{\text{new}} = \text{importance}_{\text{old}} \times \text{DECAY\_FACTOR}$$

The decay factor reduces the importance of memories over time, simulating the natural forgetting process. This ensures that less important memories are gradually phased out, making room for new information.

7 Memory Consolidation

Memories are consolidated based on their importance. If the importance exceeds a threshold, the memory is strengthened:

$$\text{importance}_{\text{new}} = \text{importance}_{\text{old}} \times \text{STRENGTHEN\_FACTOR}$$

Memories with high importance are strengthened, making them more likely to be retained in long-term memory. This process mimics how the brain prioritizes and retains important information.

8 Neuron State Update

The state of a neuron is updated based on its current state, weighted inputs, and input tensor values:

$$\text{state}_{\text{new}} = \text{state}_{\text{old}} \times 0.7 + \text{weighted\_output} \times 0.2 + \text{input\_tensor}[i\%\text{INPUT\_SIZE}] \times 0.1$$

The new state of a neuron is a weighted combination of its previous state, the weighted sum of its inputs, and the influence of the input tensor. This ensures that the neuron's state reflects both its history and the current input.

9 Neuron Output Calculation

The output of a neuron is calculated using the hyperbolic tangent (tanh) function:

$$\text{output} = \tanh(\text{state} \times \text{scale} + \text{bias})$$

The tanh function is used to introduce non-linearity into the neuron's output, ensuring that the output is bounded between -1 and 1. This non-linearity is essential for the network to model complex relationships in the data.

10 Cosine Similarity

The cosine similarity between two vectors is calculated as:

$$\text{similarity} = \frac{\sum_{i=0}^{n-1} \text{vec1}[i] \times \text{vec2}[i]}{\sqrt{\sum_{i=0}^{n-1} \text{vec1}[i]^2} \times \sqrt{\sum_{i=0}^{n-1} \text{vec2}[i]^2}}$$

Cosine similarity measures the cosine of the angle between two vectors, providing a value between -1 and 1. A value of 1 indicates perfect similarity. This metric is used to compare memory vectors and identify similar patterns.

11 Hierarchical Memory System

Larkos employs a hierarchical memory system consisting of short-term, medium-term, and long-term memory clusters. This design allows the network to prioritize and retain important information while discarding less relevant data.

11.1 Short-Term Memory

Short-term memory stores recent memories with high detail. Memories in this cluster decay quickly unless they are consolidated into medium-term memory.

11.2 Medium-Term Memory

Medium-term memory stores consolidated memories with moderate detail. Memories in this cluster are less detailed than those in short-term memory but are more stable and longer-lasting.

11.3 Long-Term Memory

Long-term memory stores highly consolidated, abstract memories. Memories in this cluster are the most stable and are retained for the longest duration.

12 Predictive Coding

Predictive coding is used to enhance the network's ability to anticipate future states based on current inputs and past experiences. The predictive coding formula is:

$$\text{prediction\_error} = \text{actual\_input} - \text{predicted\_input}$$

The network adjusts its weights and states based on the prediction error, allowing it to improve its predictive accuracy over time.

13 Advanced Neuron Management

The network employs advanced neuron management techniques to dynamically add or remove neurons based on their performance. Neurons that consistently underperform are removed, while new neurons are added to explore new patterns and improve network performance.

14 Self-Reflection System

The self-reflection system is designed to analyze the network's performance and coherence. It includes functions to detect confabulation, regenerate responses, and perform self-reflection. The main components are:

14.1 Reflection History

The system maintains historical records of reflection metrics:

$$\mathcal{H} = (\mathbf{c}_h, \mathbf{f}_h, \mathbf{s}_h, \theta_c, \theta_f, \theta_s, h_i) \tag{9}$$

where:

- $\mathbf{c}_h$ = Historical coherence scores (100-step window)
- $\mathbf{f}_h$ = Historical confidence scores
- $\mathbf{s}_h$ = Historical consistency scores
- θ_c = Coherence threshold (0.65)
- θ_f = Confidence threshold (0.7)
- θ_s = Consistency threshold (0.75)
- h_i = History index (circular buffer)

14.2 Reflection Metrics

The system computes these metrics each step:

$$\mathcal{M} = (c, f, n, s, p, r) \tag{10}$$

where:

- c = Coherence score $\in [0, 1]$
- f = Confidence score $\in [0, 1]$
- n = Novelty score $\in [0, 1]$
- s = Consistency score $\in [0, 1]$
- p = Confabulation probability $\in \{0, 1\}$
- r = Reasoning text

14.3 Coherence Analysis

The coherence score combines:

$$c = c_{\text{internal}} \cdot c_{\text{historical}} \cdot c_{\text{memory}} \tag{11}$$

$$c_{\text{internal}} = \prod_{i,j \in L} (1 - 0.05\mathbb{I}(|o_i - o_j| > 0.8)) \tag{12}$$

$$c_{\text{historical}} = 0.5 + 0.5\phi(\mathbf{n}_t, \mathbf{n}_{t-1}) \tag{13}$$

$$c_{\text{memory}} = 0.7 + 0.3\gamma(\mathbf{n}_t, \mathcal{M}_r) \tag{14}$$

where:

- ϕ = State similarity function
- γ = Memory consistency function
- $\mathcal{M}_r$ = Retrieved memory
- L = Neurons in same layer

14.4 Confidence Calculation

Neural confidence based on output-state alignment:

$$f = \frac{1}{N} \sum_{i=1}^{N} o_i (1 - |s_i|) \tag{15}$$

14.5 Novelty Detection

Divergence from historical patterns:

$$n = 1 - \frac{1}{T} \sum_{t=1}^{T} \phi(\mathbf{n}_t, \mathbf{n}_{t-\tau}) \tag{16}$$

14.6 Consistency Verification

Temporal stability measure:

$$s = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}(|o_i^{(t)} - o_i^{(t-k)}| < 0.2) \tag{17}$$

14.7 Confabulation Detection

The system detects implausible responses using:

$$p = \mathbb{I}\left[\left(\sum_{i=1}^{N} \mathbb{I}(o_i > 0.95) > 0.3N\right) \vee (c < 0.6\bar{c}_h) \vee (c < \theta_c)\right] \tag{18}$$

where $\bar{c}_h$ is the historical average coherence.

14.8 Response Regeneration

When confabulation is detected ($p = 1$):

14.9 Dynamic Parameter Adjustment

The system adapts network parameters based on reflection:

$$\alpha_{\text{learn}} \leftarrow \alpha_{\text{learn}} \times (0.8 + 0.4f) \tag{19}$$

$$\eta_{\text{noise}} \leftarrow \max(0.1, \eta_{\text{noise}}(1 - 0.2n)) \tag{20}$$

$$\beta_{\text{plasticity}} \leftarrow \begin{cases} 0.8\beta_{\text{plasticity}} & \text{if } c < \theta_c \\ \beta_{\text{plasticity}} & \text{otherwise} \end{cases} \tag{21}$$

14.10 Memory System Interaction

$$w_{\text{memory}} = 0.3 + 0.7c_{\text{memory}} \tag{22}$$

15 Knowledge Filter System

The **Knowledge Filter** categorizes inputs, evaluates their relevance, and determines how they should be integrated into memory. It ensures only meaningful information is retained while filtering out noise.

15.1 Key Components

- **Knowledge Categories** A set of predefined categories (e.g., Pattern Recognition, Prediction, Optimization) with associated feature vectors:

  $$\mathcal{K} = \{K_1, K_2, \ldots, K_n\}, \quad K_i = (\mathbf{f}_i, w_i, c_i)$$

  where:

  - $\mathbf{f}_i$ = feature vector (dimension d)
  - w_i = importance weight
  - c_i = confidence score

- **Problem Instance Tracking** Records past experiences as:

  $$P_j = (\mathbf{x}_j, t_j, s_j, K_j)$$

  where $\mathbf{x}_j$ = input features, t_j = timestamp, s_j = success rate, and K_j = associated category.

- **Similarity Matrix** Computes pairwise category similarities:

  $$S_{ij} = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\|\|\mathbf{f}_j\|}$$

15.2 Core Functions

1. **Input Categorization** Assigns input $\mathbf{x}$ to category K_i if:

$$\arg\max_i \left(S(\mathbf{x}, \mathbf{f}_i)\right) > \theta_{\text{sim}}$$

where θ_{sim} is a similarity threshold.

2. **Dynamic Re-weighting** Updates importance w_i based on usage and success:

$$w_i^{(t+1)} = \alpha w_i^{(t)} + (1-\alpha)\left(\frac{\text{usage}_i}{\text{max\_usage}} + s_i\right)$$

3. **Memory Integration** Strengthens memories linked to high-importance categories:

$$\text{Memory Strength} \propto \sum_{K_i \in \mathcal{K}} w_i \cdot c_i$$

16 Self-Identity System

The **Self-Identity System** maintains a coherent representation of the agent's core values, beliefs, and behavioral patterns.

16.1 Key Components

- **Core Values** A vector $\mathbf{v} \in \mathbb{R}^m$ representing fundamental principles:

$$v_i^{(t+1)} = v_i^{(t)} + \eta \cdot \text{consistency}(\mathbf{v}, \mathbf{b})$$

- **Belief System** Dynamic beliefs $\mathbf{b} \in \mathbb{R}^n$ updated via:

$$b_i^{(t+1)} = b_i^{(t)} + \gamma \cdot (\text{memory\_influence} + \text{experience\_influence})$$

- **Identity Markers** Stable traits $\mathbf{m} \in \mathbb{R}^p$ adjusted by:

$$m_i^{(t+1)} = (1-\beta) m_i^{(t)} + \beta \cdot (\mathbf{v} \cdot \mathbf{b})$$

16.2 Consistency Metrics

$$\text{Value-Belief Consistency} = 1 - \frac{1}{n}\sum_{i=1}^{n} |v_i - b_i|$$

$$\text{Temporal Coherence} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(\|\mathbf{m}^{(t)} - \mathbf{m}^{(t-1)}\| < \epsilon)$$

17 Integration

The two systems interact via:

$$\text{Knowledge Integration Weight} = 0.3 + 0.7 \cdot \text{Identity Coherence}$$

18 Integration and Interaction

18.1 Memory System Interaction

The knowledge filter system influences memory operations by strengthening memories based on their importance. The self-identity system updates its components based on memories and experiences.

Metric	Role
Category Confidence (c_i)	Modulates memory retention
Identity Coherence	Adjusts learning rate
Belief-Value Alignment	Filters conflicting inputs

Table 1: Cross-system interactions

18.2 Emotional System Interaction

The emotional system modulates neural processing based on emotional states. Emotions influence decision-making and learning, providing affective modulation of cognitive processes.

18.3 Predictive Coding Enhancement

The knowledge filter system enhances predictive coding by improving the network's ability to anticipate future states. The self-identity system ensures that predictions are consistent with the network's identity and values.

19 Meta-Controller System

The Meta-Controller operates as the executive control mechanism for larkos, managing global resource allocation and learning strategies.

19.1 Core Components

$$\mathcal{MC} = (\alpha_{\text{meta}}, \epsilon, \mathcal{R}, \mathbf{w}_r, \mathbf{h}_r) \tag{23}$$

where:

- α_{meta} = Meta-learning rate
- ϵ = Exploration factor
- $\mathcal{R}$ = Set of neural regions
- $\mathbf{w}_r$ = Region importance weights
- $\mathbf{h}_r$ = Region performance history

19.2 Dynamic Adaptation

The controller updates region importance using:

$$w_r^{(t+1)} = w_r^{(t)} + \alpha_{\text{eff}} \Delta_r (1 + \epsilon_{\text{dyn}}) c_r \tag{24}$$

where:

$$\begin{aligned}
\alpha_{\text{eff}} &= \alpha_{\text{meta}}(1 + \alpha_{\text{mc}})(1 - L_c) \\
\epsilon_{\text{dyn}} &= \epsilon(1 + \tau_p)C \\
\Delta_r &= \text{Performance delta for region } r \\
c_r &= \text{Context relevance score}
\end{aligned}$$

20 Meta-Cognition System

The Meta-Cognition system provides self-monitoring and reflection capabilities.

20.1 Key Metrics

$$\mathcal{M} = (C, \alpha, L, E, c) \tag{25}$$

where:

- C = Confidence level $\in [0, 1]$
- α = Adaptation rate
- L = Cognitive load $\in [0, 1]$
- E = Error awareness
- c = Context relevance

20.2 Metric Calculations

$$C = \frac{1}{1 + \sigma_p^2} \tag{26}$$

$$L = \frac{1}{1 + e^{-(0.4H_a + 0.3C_w + 0.3C_t)}} \tag{27}$$

$$E = \frac{\bar{E} + E_{\max}}{2} \tag{28}$$

$$c = \frac{\sum w_r p_r}{|\mathcal{R}|} \tag{29}$$

where:

- σ_p^2 = Performance variance
- H_a = Activation entropy
- C_w = Weight complexity
- C_t = Temporal complexity

21 Emotional System

The emotional system provides affective modulation of the network's cognitive processes. Key components include:

21.1 Emotion Representation

Emotions are represented as continuous values with intensity, decay rates, and influence factors:

$$\text{emotion} = \{\text{intensity}, \text{decay\_rate}, \text{influence\_factor}, \text{threshold}\}$$

21.2 Emotion Triggering

Emotions are triggered based on network states and performance:

$$\text{trigger\_strength} = f(\text{error\_rate}, \text{social\_context}, \text{cooperation\_level})$$

$$\text{intensity}_{\text{new}} = \text{intensity}_{\text{old}} \times (1 - \text{decay\_rate}) + \text{trigger\_strength}$$

21.3 Emotional Bias

Emotions influence neural processing through a bias term:

$$\text{emotional\_bias} = \sum (\text{emotion\_intensity} \times \text{influence\_factor}) \times \text{cognitive\_impact}$$

21.4 Valence and Arousal

The system calculates overall affective state:

$$\text{valence} = \text{positive\_emotions} - \text{negative\_emotions}$$

$$\text{arousal} = \sum \text{high\_activation\_emotions}$$

21.5 Emotion Representation and Dynamics

The emotional system now represents emotions as multi-dimensional vectors, incorporating valence, arousal, dominance, complexity, and temporal depth. The system also tracks momentum for smooth emotional transitions.

$$\text{EmotionVector} = \{\text{valence}, \text{arousal}, \text{dominance}, \text{complexity}, \text{temporal\_depth}, \text{duration\_steps}, \text{stability}, \text{momentum}\} \tag{30}$$

- **Valence**: Ranges from -1.0 (negative) to 1.0 (positive)
- **Arousal**: Ranges from 0.0 (calm) to 1.0 (excited)
- **Dominance**: Ranges from 0.0 (submissive) to 1.0 (dominant)
- **Complexity**: Represents emotional nuance, ranging from 0.0 to 1.0
- **Temporal Depth**: Measures emotional persistence over time
- **Stability**: Reflects emotional resilience, ranging from 0.0 to 1.0
- **Momentum**: Tracks the rate of change for valence, arousal, and dominance

21.6 Emotional Attractors

The system introduces emotional attractors, which are stable emotional states that influence the current emotional trajectory. Each attractor has:

- A center point (target emotional state)
- Basin strength (pull toward the attractor)
- Visit count and average duration (historical engagement)
- Linked attractors for emotional transitions

The system computes the distance between the current emotional state and attractors using a weighted Euclidean metric:

$$\text{distance} = \sqrt{(\Delta\text{valence})^2 + (\Delta\text{arousal})^2 + (\Delta\text{dominance})^2 + 0.5(\Delta\text{complexity})^2 + 0.3(\Delta\text{temporal\_depth})^2} \tag{31}$$

The nearest attractor is selected based on distance and context alignment.

21.7 Attachment Bonds

The system models attachment bonds between the agent and external entities:

$$\text{AttachmentBond} = \{\text{entity\_id}, \text{entity\_name}, \text{attachment\_strength}, \text{trust}, \text{familiarity}, \text{dependency}, \text{care\_investment}, \text{loss\_cost} \tag{32}$$

21.8 Emotion-Embedding Integration

Emotional states influence the embedding space through:

$$\text{embeddings}[i] = \text{embeddings}[i]\times\text{valence\_scale}\times\text{arousal\_scale}+\text{affective\_bias}\times\text{emotional\_weight}+\text{attractor\_influence} \tag{33}$$

Where:

- valence_scale = 1.0 + current_state.valence × 0.3
- arousal_scale = 1.0 + current_state.arousal × 0.2
- affective_bias = affective_embeddings[i]
- emotional_weight = (current_state.valence × 0.4 + current_state.arousal × 0.3 + current_state.dominance × 0.3) × plasticity
- attractor_influence = context_weights[i] × basin_strength × 0.2 (if in attractor)

21.9 Affective Complexity

The system calculates affective complexity based on:

$$\text{complexity\_target} = 0.2+\text{unique\_attractors}\times0.08+\sum\text{conflict\_history}\times0.1+\text{subconscious\_influence}\times0.15 \tag{34}$$

21.10 Emotion Momentum Update

Emotional states transition smoothly using momentum:

$$\begin{aligned}
\text{momentum}[0] &= \text{momentum}[0] \times 0.9 + (\text{target.valence} - \text{current.valence}) \times \text{dt}\\
\text{momentum}[1] &= \text{momentum}[1] \times 0.9 + (\text{target.arousal} - \text{current.arousal}) \times \text{dt}\\
\text{momentum}[2] &= \text{momentum}[2] \times 0.9 + (\text{target.dominance} - \text{current.dominance}) \times \text{dt}\\
\text{current.valence} &= \text{current.valence} + \text{momentum}[0]\\
\text{current.arousal} &= \text{current.arousal} + \text{momentum}[1]\\
\text{current.dominance} &= \text{current.dominance} + \text{momentum}[2]
\end{aligned} \tag{35}$$

21.11 Attachment Bond Dynamics

Attachment bonds evolve through interactions:

$$\begin{aligned}
\text{attachment\_strength} &= \text{attachment\_strength} \times 0.95 + \text{attachment\_target} \times 0.05\\
\text{attachment\_target} &= \text{familiarity} \times 0.3 + \text{trust} \times 0.25 + \text{emotional\_resonance} \times 0.2 + \text{care\_investment} \times 0.15 + (1 - \text{conf}\\
\text{loss\_cost} &= \text{attachment\_strength} \times \text{care\_investment} \times (1 + \text{dependency}) \times 0.5
\end{aligned} \tag{36}$$

21.12 Emotional Trajectory Simulation

The system simulates emotional trajectories over time:

$$\text{simulateEmotionalTrajectory}(\text{steps}, \text{context}) \rightarrow \{\text{emotion\_history}, \text{attractor\_transitions}, \text{complexity\_changes}\} \tag{37}$$

21.13 Integration with Identity System

Attachment bonds influence identity core values:

$$\text{identity\_core\_values}[i] += \text{care\_investment} \times \text{bond\_influence} \times 0.1 \tag{38}$$

21.14 Predictive Commitment

The system adjusts predictive commitment based on prediction errors:

$$\text{predictive\_commitment\_weight} = \text{predictive\_commitment\_weight} - \text{prediction\_error} \times 0.1 \times \text{predictive\_commitment\_weight} \tag{39}$$

22 Imagination System

The imagination system enables mental simulation of scenarios before action.

22.1 Scenario Generation

Scenarios are generated by perturbing current states:

$$\text{scenario\_vector} = \text{current\_state} + \mathcal{N}(0, \text{divergence\_factor})$$

22.2 Scenario Simulation

Scenarios are simulated through forward passes:

$$\text{simulated\_state}_{t+1} = f(\text{simulated\_state}_t, \text{input} + \epsilon)$$

where ϵ is noise proportional to divergence factor.

22.3 Plausibility Evaluation

Scenarios are evaluated against memory:

$$\text{plausibility} = \text{similarity}(\text{scenario}, \text{memory}) \times (1 - \text{divergence})$$

22.4 Outcome Blending

The network blends imagined outcomes into its state:

$$\text{state}_{\text{new}} = (1 - \alpha)\text{state}_{\text{current}} + \alpha\text{state}_{\text{imagined}}$$

23 Social System

The social system enables modeling of other agents and social interactions.

23.1 Person Modeling

The system maintains models of other agents:

$$\text{person\_model} = \{\text{traits}, \text{trust}, \text{prediction\_accuracy}\}$$

23.2 Behavior Prediction

Behavior is predicted based on models and context:

$$\text{predicted\_behavior} = \sum w_i \times \text{observed\_behavior}_i + (1 - w_i) \times \text{trait\_baseline}$$

23.3 Social Learning

Social capabilities improve through experience:

$$\text{empathy}_{\text{new}} = \text{empathy}_{\text{old}} + \eta(1 - \text{emotion\_prediction\_error})$$

23.4 Negotiation

The system finds compromise solutions:

$$\text{compromise} = \frac{\text{self\_goal} \times \text{self\_weight} + \text{other\_goal} \times \text{other\_weight}}{\text{self\_weight} + \text{other\_weight}}$$

24 Validating System

The validating system provides robust error detection and recovery mechanisms to ensure system stability and prevent catastrophic failures. It operates through several key components:

24.1 Memory Validation

The system implements comprehensive memory validation with:

- **Segmentation Fault Protection**: Uses signal handlers to catch and recover from memory access violations
- **Memory Region Checks**: Validates pointer accessibility before operations
- **Boundary Verification**: Ensures all memory operations stay within allocated bounds

24.2 Error Detection Mechanisms

$$\text{valid} = \begin{cases} 1 & \text{if ptr} \neq \text{NULL} \wedge \text{size} > 0 \wedge \text{isAccessible(ptr)} \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

24.3 System Components Validation

The validator checks all critical subsystems:

1. Working Memory System
2. Meta Controller
3. Performance Metrics
4. Motivation System
5. Reflection Parameters
6. Identity System
7. Knowledge Filter
8. Metacognition System
9. Social System
10. Emotional System

24.4 Recovery Mechanisms

The system implements multiple recovery strategies:

- **Automatic Correction**: Resets invalid values to safe defaults
- **Memory Reallocation**: Recreates corrupted data structures
- **Emergency Backup**: Saves system state before critical operations
- **Health Monitoring**: Tracks system stability metrics over time

24.5 Health Metrics Tracking

The system maintains comprehensive health statistics:

$$H = \left(\frac{S}{T}, \frac{F}{T}, \bar{t}, M_{\text{corr}}\right) \tag{41}$$

where:

- S = Successful checks
- F = Failed checks
- T = Total checks
- $\bar{t}$ = Average check time
- M_{corr} = Correction count

24.6 Floating Point Protection

Special handlers detect and recover from numerical errors:

- Division by zero
- Overflow/underflow
- Invalid operations
- Inexact results

24.7 System Stabilization

When instability is detected, the system:

1. Logs current state
2. Creates emergency backup
3. Resets volatile components
4. Reinitializes critical subsystems

24.8 Validation Process

The complete validation workflow:

1. Install signal handlers
2. Check memory regions
3. Validate subsystem structures
4. Verify numerical values
5. Correct invalid states
6. Update health metrics
7. Generate stability report

24.9 Performance Impact

The validation system operates with minimal overhead:

- Average check time: 0.1-0.5ms
- Memory usage: ¡1MB
- Recovery time: 10-100ms for most errors

24.10 Integration

The validator interacts with other systems through:

- Memory system for allocation tracking
- Performance metrics for error logging
- Meta-controller for adaptive validation frequency
- Reflection system for error analysis