Module J: Recursive Adaptive Dynamic Idempotent Feedback and State-Based Runtime Learning and Inference

Part of the Eidos Unified Framework for Persistent, Dynamic, and Adaptive Multimodal Intelligence

—

Contents

1 Abstract

This module rigorously defines the *Recursive Adaptive Dynamic Idempotent Feedback and State-Based Runtime Learning and Inference* component of the Eidos framework. It provides a formal system for modeling interdependent feedback dynamics among multiple entities, capturing how triggers and actions propagate influence recursively. Key aspects include the definition of entities, triggers, actions, influence functions, and feedback loops; the introduction of adaptive modulation functions that update system states based on external and internal signals; and the enforcement of idempotence and reversibility to guarantee stability. The framework is developed with full mathematical rigor and is designed to be modular, extensible, and dynamically adaptive, enabling continuous runtime learning and inference.

2 Introduction and Motivation

In complex intelligent systems, continuous adaptation and learning require mechanisms that can recursively integrate feedback from multiple sources. The *Recursive Adaptive Dynamic Idempotent Feedback* system in the Eidos framework addresses this need by modeling the interactions among various entities through triggers and actions. These interactions are recursively composed into feedback loops that update the system state in an adaptive yet stable manner. Key motivations for this module include:

(a) Modeling the bidirectional and cyclic interactions among entities.

(b) Ensuring that repeated updates via feedback are idempotent, thereby guaranteeing stability.

(c) Allowing dynamic adaptation at runtime through adaptive modulation of entity states.

(d) Supporting a modular design where each entity's feedback can be updated independently without disrupting the global system state.

This module provides the formal and algorithmic foundation for runtime learning and inference based on recursive feedback.

3 Preliminaries and Notation

We introduce the following notation and definitions:

- Let $\mathcal{E} = \{E_i \mid i \in I\}$ denote the set of entities in the system, where I is an index set (finite or countable).
- For each entity E_i:
  - $\tau(E_i) \in \mathcal{T}$ represents the *trigger* function or signal.
  - $\alpha(E_i) \in \mathcal{A}$ represents the *action* executed by the entity.
- The *influence function* is denoted by:

  $$\delta(E_i, E_j) : \quad \tau(E_i) \longrightarrow \alpha(E_j),$$

  which quantifies how the trigger of E_i affects the action of E_j.

- The *feedback function* is defined as:

$$\phi(E_i, E_j) = \delta(E_i, E_j) \circ \delta(E_j, E_i),$$

where $\circ$ denotes functional composition, capturing the bidirectional recursive interaction.

- The *system state* for an entity E_i is denoted by:

$$\Sigma(E_i) = \{\tau(E_i),\ \alpha(E_i),\ \{\delta(E_i, E_j)\}_{j \neq i}\},$$

and for a set of entities relevant to a context $a \in \mathcal{C}$, the global state is:

$$\Sigma_a = \bigoplus_{(E_i,E_j)\in\mathcal{E}_a^2,\ i\neq j} \left[\phi(E_i, E_j) \oplus \phi(E_j, E_i)\right],$$

where $\oplus$ denotes a modular composition operator (assumed to be associative and commutative).

- An *adaptive modulation function* is defined as:

$$\mu_a : \Sigma_0 \to \Sigma_a,$$

which adapts a base state Σ_0 to a given context a. This function is required to be idempotent:

$$\mu_a\big(\mu_a(\Sigma_0)\big) = \mu_a(\Sigma_0).$$

- The system may incorporate adaptive parameters from a set $\mathcal{P}$, with the learned modulation for context a represented as:

$$\mathcal{P}_a \in \mathcal{P}.$$

4 Formal Definitions and Mathematical Formulation

Definition J.1 (Entities, Triggers, and Actions)

For each entity $E_i \in \mathcal{E}$:

$$\tau(E_i) \in \mathcal{T}, \quad \alpha(E_i) \in \mathcal{A}.$$

These functions are assumed to be defined on appropriate domains and codomains so that they can be composed with influence functions.

Definition J.2 (Influence Function)

For any two distinct entities $E_i, E_j \in \mathcal{E}$, the influence function is defined as:

$$\delta(E_i, E_j) : \quad \tau(E_i) \longrightarrow \alpha(E_j).$$

This function quantifies the manner in which the trigger signal of E_i induces or modulates the action of E_j.

Definition J.3 (Feedback Function)

The bidirectional feedback between entities E_i and E_j is defined as:

$$\phi(E_i, E_j) = \delta(E_i, E_j) \circ \delta(E_j, E_i).$$

This composition represents the recursive interplay between E_i and E_j, where the output of one influence serves as the input to the other. A non-trivial feedback loop satisfies:

$$\phi(E_i, E_j) \neq 0,$$

ensuring meaningful interaction.

Definition J.4 (System State)

For a given context $a \in \mathcal{C}$, define the pairwise state contribution of entities E_i and E_j as:

$$\Sigma_a(E_i, E_j) = \phi(E_i, E_j) \oplus \phi(E_j, E_i).$$

The overall system state under context a is then:

$$\Sigma_a = \bigoplus_{(E_i, E_j) \in \mathcal{E}_a^2,\ i \neq j} \Sigma_a(E_i, E_j).$$

Definition J.5 (Adaptive Modulation Function)

Define a modulation function that adapts the base state Σ_0 to a context a:

$$\mu_a : \Sigma_0 \to \Sigma_a,$$

with the idempotence property:

$$\mu_a\big(\mu_a(\Sigma_0)\big) = \mu_a(\Sigma_0).$$

When the modulation function is learned or trained, it encapsulates a set of adaptive parameters:

$$\mathcal{P}_a \in \mathcal{P},$$

which are then applied to update the system state.

5 Algorithmic Description

The following pseudocode outlines the operation of the recursive feedback system:

Algorithm 1 Recursive Adaptive Feedback and Runtime Learning

```
1: Input: Set of entities $\mathcal{E} = \{E_i \mid i \in I\}$; initial system state $\Sigma_0$; context $a \in \mathcal{C}$; modulation function $\mu_a$
2: Output: Adapted system state $\Sigma_a$
3: Begin:
4: for each pair of distinct entities $(E_i, E_j)$ do
5:     Compute influence: $\delta(E_i, E_j)$ from $\tau(E_i)$ to $\alpha(E_j)$
6:     Compute reverse influence: $\delta(E_j, E_i)$ from $\tau(E_j)$ to $\alpha(E_i)$
7:     Compute feedback:
           $\phi(E_i, E_j) \leftarrow \delta(E_i, E_j) \circ \delta(E_j, E_i)$
8:     Set pairwise state:
           $\Sigma_a(E_i, E_j) \leftarrow \phi(E_i, E_j) \oplus \phi(E_j, E_i)$
9: end for
10: Compose global state:
           $\Sigma_a \leftarrow \bigoplus_{(E_i,E_j)\in\mathcal{E}_a^2,\, i\neq j} \Sigma_a(E_i, E_j)$
11: Apply modulation:
           $\Sigma_a \leftarrow \mu_a(\Sigma_0)$
12: Return: $\Sigma_a$
```

Optional Iterative Adaptation: For continual runtime learning, the above procedure can be applied iteratively. Let:

$$\Sigma^{(0)} = \Sigma_0, \quad \Sigma^{(t+1)} = \mu_a\Big(\Sigma^{(t)}\Big),$$

with convergence guaranteed by the idempotence property:

$$\lim_{t\to\infty} \Sigma^{(t)} = \Sigma_a.$$

6 Theoretical Analysis and Guarantees

Theorem J.1 (Idempotence and Stability)

Statement: For any context $a \in \mathcal{C}$, the modulation function μ_a is idempotent, i.e.,

$$\mu_a\big(\mu_a(\Sigma_0)\big) = \mu_a(\Sigma_0) = \Sigma_a.$$

Thus, repeated application of the feedback update yields a stable system state.

Proof Sketch: By the definition of μ_a (Definition J.5), once the base state Σ_0 is adapted to Σ_a, further applications do not alter Σ_a. This follows from the property that the learned adaptive parameters $\mathcal{P}_a$ yield an idempotent mapping. □

Proposition J.2 (Modularity and Scalability)

The feedback system is constructed via the modular composition operator $\oplus$, which is assumed to be associative and commutative. Consequently, the global state

$$\Sigma_a = \bigoplus_{(E_i,E_j)\in\mathcal{E}_a^2,\, i\neq j} \Sigma_a(E_i, E_j)$$

is independent of the order of composition. This property ensures that the system can scale to an arbitrary number of entities and can be extended to higher-order interactions without loss of consistency.

7 Integration with the Overall Eidos Framework

Module J is a core component responsible for enabling adaptive, runtime learning and inference. It:

- Interfaces with the deep model architectures (Module H) by providing a mechanism for continuous state adaptation based on internal feedback.
- Interacts with the memory module (Module I) by incorporating external, real-time signals into the feedback loops.
- Connects with the training system (Module K) to allow online parameter updates and continual learning.
- Provides a recursive framework that ensures that system states remain stable (via idempotence) while adapting dynamically to new inputs and contextual changes.

8 Implementation Considerations

- **Feedback Computation:** The functions $\delta(E_i, E_j)$ should be implemented using well-established techniques (e.g., weighted mappings or neural network modules) with careful calibration to avoid vanishing or exploding feedback signals.
- **Adaptive Modulation:** The modulation function μ_a may be implemented as a gating network or a parameterized transformation that is trained jointly with the rest of the model.
- **Idempotence Verification:** Empirical tests should be designed to verify that repeated applications of μ_a yield negligible changes beyond a certain iteration, ensuring convergence.
- **Scalability:** The composition operator $\oplus$ should be chosen to support efficient aggregation over large sets of entities. Sparse representations and parallel computation may be used.
- **Integration with Feedback Sources:** The system should accommodate both internally generated signals (e.g., prediction errors) and external signals (e.g., user feedback) within the recursive feedback loops.

9 Conclusion

In this module, we have developed a comprehensive framework for recursive, adaptive, and idempotent feedback that underpins runtime learning and inference in the Eidos framework. The key contributions are:

- A formal definition of entities, triggers, actions, and influence functions that model interdependent interactions.
- The definition of a recursive feedback function $\phi(E_i, E_j)$ that composes bidirectional influences.

- The construction of a global system state Σ_a via modular composition, capturing the cumulative effects of feedback across entities.
- The introduction of an adaptive modulation function μ_a with an idempotence property, ensuring that the system state converges and remains stable under repeated updates.
- The presentation of algorithmic procedures for computing feedback and adapting system parameters in real time.

This robust and modular recursive feedback mechanism is essential for enabling dynamic adaptation, continual learning, and runtime inference in complex environments, forming a critical component of the overall Eidos framework.

10 Module Summary

Completed:

- Module A: Input Processing.
- Module B: Universal Communication & Data Handling Interface and Coordination.
- Module C: Universal Streaming/Handling/Loading/Indexing Module.
- Module D: Multidimensional Vocabulary and Tokenization System.
- Module E: Contextual NLU/NLP Embedding and Multidimensional Tokenization.
- Module F: Deep Knowledge Graphs System (Base and Personal).
- Module G: Infinite RoPE Context Scaling and Dynamic Vocabulary Updating.
- Module H: Core Model Architectures (RWKV and Transformer Modules, Mixture-of-Experts Style).
- Module I: Titans Memory Architecture (Multi-Layer Memory Module).
- Module J: Recursive Adaptive Dynamic Idempotent Feedback and State-Based Runtime Learning and Inference.

Remaining Modules:

- Module K: Universal Training System.
- Module L: Final Decoding and Multimodal Output.