Recursive Goal Selection and Arbitration in Cognitive Architectures

Klea
Independent AI Cognitive Architect
KleaSCM@gmail.com

June 2, 2025

Abstract

This paper presents a formal, recursive model of goal selection and arbitration in artificial cognitive architectures. We define a structured framework for representing goal states, arbitration mechanisms, recursive dependencies, and adaptive prioritization in dynamic environments. The model integrates utility-based evaluation, recursive arbitration loops, and state-aware conflict resolution. Intended applications include autonomous decision systems, AGI scaffolds, and adaptive planning modules.

Contents

1 Problem Definition and Scope **6**
1.1 Introduction 6
1.2 Formal Problem Statement 6
1.3 Scope of the Model 6
1.4 Role of Recursion 7
1.5 Assumptions and Constraints 7

2 Mathematical Foundations 8
2.1 Cognitive State Space 8

3 Goal Representation 9
3.1 Goal Encoding 9
3.2 Goal Hierarchies and Dependencies 10
3.3 Goal Priority and Urgency 11

4 Arbitration Mechanisms 12
4.1 Formal Models of Arbitration 12
4.1.1 Game-Theoretic Arbitration: Nash Equilibrium 12
4.2 Conflict Resolution Frameworks 13
4.3 Stability and Convergence Criteria 14
4.4 Refined Contraction Conditions for Arbitration Dynamics . . . 15
4.5 Characterization of Equilibria in Multi-Goal Systems 15
4.6 Examples of Arbitration Dynamics and Convergence 16

5 Lipschitz Continuity of Arbitration Operators 17
5.1 Lipschitz Constant of the Softmax Operator 17
5.2 Lipschitz Constant for Linear Utility Mappings 18
5.3 Contraction Condition for Composite Arbitration 18
5.4 Summary of Stability Analysis 19

6 Recursion in Goal Selection 19
6.1 Recursive Goal Evaluation 19
6.2 Recursive Arbitration Dynamics 20
6.3 Fixed Points and Attractors in Recursive Decision Dynamics . 20
6.4 Contraction Conditions for Combined Recursive Maps 21
6.5 Examples of Recursive Arbitration Stability 23
6.5.1 Example 1: Softmax Arbitration with Linear Utilities . 23
6.5.2 Example 2: Softmax Arbitration with α-dependent Utilities 24
6.5.3 Example 3: Noisy Memory Arbitration with Smoothing 24
6.5.4 Example 4: Nonlinear Utilities with Bounded Curvature 25
6.5.5 Example 5: Adversarial Goal Perturbation 25
6.6 Summary of Recursive Arbitration Stability 26

7 Dynamic Update Rules 27
7.1 Recursive Arbitration as Base Dynamics 27
7.2 External Input Integration 27
7.3 Trait-Driven Modulation 28
7.4 Full Recursive Update Schema 28
7.5 Illustrative Examples and Simulation Scenarios 28
7.6 Summary of Dynamic Update Rules 30

8 Stochasticity and Noise in Arbitration Dynamics 30
8.1 Stochastic Utility Functions 30
8.2 Probabilistic Arbitration Mechanisms 31
8.3 Stochastic Differential Equation Formulation 31
8.4 Effect of Noise on Convergence and Adaptability 32
8.5 Summary of Stochastic Arbitration Dynamics 33
8.6 Concrete Examples of Stochastic Arbitration 33
8.7 Trait-Sensitivity Experiments: Impact on Arbitration Dynamics 35
8.7.1 Trait Parameters 36
8.7.2 Simulation Setup 36
8.7.3 Results and Interpretation 37
8.7.4 Theoretical Implication 37
8.7.5 Conclusion 37
8.8 Summary: Stochasticity and Trait-Modulated Arbitration . . . 38
8.9 Implementation Architecture 39
8.9.1 Data Structures for Recursive Goal Arbitration 39
8.9.2 Recursive Arbitration Algorithms 39
8.9.3 Complexity Considerations 40
8.9.4 Integration with Cognitive Modules 40
8.9.5 Advanced Implementation Features 40
8.9.6 Summary of Implementation Architecture 41
8.10 Performance Metrics 41
8.10.1 Arbitration Stability 41
8.10.2 Goal Switching Entropy 42
8.10.3 Utility Convergence Rate 42
8.10.4 Trait Responsiveness Index 43
8.10.5 Latency and Real-Time Performance 43
8.10.6 Task Success Rate 43
8.10.7 Summary of Metrics 44

9 Simulation & Analysis **46**
9.1 Test Scenarios for Recursive Goal Arbitration 46
9.1.1 Scenario 1: Competing Urgent vs. Important Goals . . 47
9.1.2 Scenario 2: Oscillating Environmental Conditions . . . 48
9.1.3 Scenario 3: Goal Injection Perturbation (Shock Test) . 49
9.1.4 Scenario 4: Memory-Interference Induced Goal Collapse 50
9.2 Summary 51

10 Stability Analysis of Recursive Cognitive Goal States via Lyapunov Theory **52**
10.1 Introduction 52
10.2 Preliminaries and Definitions 52
10.3 Linearization and Local Stability 52
10.4 Lyapunov Stability: Definitions and Theorems 53
10.5 Constructing Lyapunov Functions for Recursive Cognitive Systems 53
10.6 Extensions to Nonlinear and Global Stability 54
10.7 Practical Considerations in Cognitive Architectures 55
10.8 Computational Algorithms for Lyapunov-Based Stability Certification 56
10.8.1 Polynomial Lyapunov Functions and SOS Programming 57
10.8.2 Formulation as Semidefinite Program (SDP) 58
10.8.3 SOS Conditions for Global Asymptotic Stability 58
10.8.4 Barrier Certificates for Invariant Sets 58
10.8.5 Extensions to Uncertain and Parametric Systems . . . 59
10.8.6 Summary and Implementation Notes 59
10.9 Summary and Implications 61

11 Multi-Agent Interactions **61**
11.1 Agent Architecture in Multi-Agent Context 61
11.2 Recursive Payoff Integration Across Agents 62
11.3 Recursive Nash Equilibrium Conditions 62
11.4 Cooperative Attractor Merging 62
11.5 Competition via Attractor Divergence 63
11.6 Trait Feedback and Coupled Learning 63

12 Learning Mechanisms for Goal Preference Adaptation 67
12.1 Parameterization of Goal Fields 67
12.2 Objective Function and Gradient Formulation 67
12.3 Temporal Difference Error and Value Approximation 68
12.4 Trait Modulation and Meta-Learning 68
12.5 Algorithm: Recursive Learning for Goal Preference Adaptation 69
12.6 Stability Analysis 69
12.7 Discussion 70

13 Metacognitive Goal Revision and Self-Monitoring 70
13.1 Introduction and Motivation 70
13.2 Metacognitive State Representation 70
13.3 Metacognitive Monitoring Function Φ 71
13.4 Metacognitive Goal Revision Operator 72
13.5 Adaptive Learning Parameter Updates 73
13.6 Stability Analysis of Coupled Dynamics 73
13.7 Algorithmic Formalization 74
13.8 Interpretation and Practical Implications 74

14 Metacognitive Goal Revision and Self-Monitoring 75
14.1 Recursive Update Model 75
14.2 Assumptions and Properties 76
14.3 Convergence Result 76
14.4 Proof Sketch 77
14.5 Lyapunov Stability Analysis 77
14.6 Additional Examples 78
14.7 Remarks and Extensions 78

15 Conclusion 78

1 Problem Definition and Scope

1.1 Introduction

Goal selection and arbitration are core components of cognitive architectures, enabling agents to navigate complex environments, balance competing objectives, and make coherent decisions across time. This module governs how cognitive agents identify, prioritize, and execute goals from a dynamic set of possibilities, often under uncertainty and resource constraints.

1.2 Formal Problem Statement

Let $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$ denote a dynamic set of goals available to the agent. Let $S(t)$ represent the internal cognitive state of the agent at time t, and $E(t)$ represent the external environment or perceptual input.

We define the goal selection function:

$$\pi : (S(t), E(t)) \to \mathcal{G}_{\text{active}}(t)$$

where $\mathcal{G}_{\text{active}}(t) \subseteq \mathcal{G}$ denotes the currently activated or selected goals.

Arbitration determines which goal (or combination thereof) governs the agent's behavior at time t, denoted:

$$\alpha : \mathcal{G}_{\text{active}}(t) \to g^*(t)$$

where $g^*(t)$ is the arbitrated goal — the one to be pursued or given highest execution priority.

1.3 Scope of the Model

This document aims to formalize the entire process of recursive goal selection and arbitration, including:

- Representation of goals, priorities, and conflicts.
- Arbitration mechanisms and utility-based resolution.
- Recursive structures where goals depend on evaluation of other goals or prior arbitration results.

- Mathematical rigor with respect to system dynamics, stability, and convergence.
- Integration with cognitive traits and adaptive modulation.

1.4 Role of Recursion

Recursion enters in two critical layers:

1. **Recursive goal generation:** Goals can spawn or modify subgoals, forming a dependency graph or hierarchy.
2. **Recursive arbitration:** The process of arbitration itself is dynamically influenced by prior arbitration outcomes, evolving cognitive state, or even metacognitive goals (e.g., "decide how to decide").

The aim is to produce a robust, generalizable recursive formulation that scales across agent types, from symbolic AI to neural or hybrid cognitive architectures.

1.5 Assumptions and Constraints

We assume the following constraints for tractability and clarity:

- The cognitive state space $S(t)$ and environment $E(t)$ are both finite-dimensional and measurable.
- Goal utility functions are real-valued, bounded, and Lipschitz-continuous with respect to state variables.
- Time can be modeled either continuously ($t \in \mathbb{R}_{\geq 0}$) or discretely ($t \in \mathbb{N}$), depending on context.
- Cognitive resource constraints (e.g., attention, memory) are expressible as bounded operators.

2 Mathematical Foundations

2.1 Cognitive State Space

We define the cognitive state space $\mathcal{S}$ as a structured and recursively dynamic manifold that encodes all relevant internal agent variables contributing to goal selection and arbitration processes. Formally:

$$\mathcal{S} = \mathcal{M} \times \mathcal{A} \times \mathcal{T} \times \mathcal{I} \times \mathcal{F}$$

where:

- $\mathcal{M}$: Memory subspace (episodic, semantic, procedural)
- $\mathcal{A}$: Attention vector field
- $\mathcal{T}$: Trait/parameter vector (e.g., openness, urgency sensitivity)
- $\mathcal{I}$: Intentional trajectory space (active goal-related traces)
- $\mathcal{F}$: Affective field (emotions, valence, arousal)

Each subspace $\mathcal{X}_i$ is a high-dimensional (potentially infinite-dimensional) manifold or vector space, possibly evolving on different timescales.

State Evolution The agent's cognitive state evolves according to a transition function:

$$s_{t+1} = \Phi_t(s_t, e_t)$$

where $s_t \in \mathcal{S}$ is the cognitive state at time t, $e_t \in \mathcal{E}$ is the external input (environmental or social stimulus), and Φ_t is a possibly nonlinear transition operator. Crucially, Φ_t may itself depend on prior cognitive evaluations — encoding recursion:

$$\Phi_t = f(\Phi_{t-1}, \nabla_s J(s_t), e_t)$$

where $J(s_t)$ is a utility or valuation function defined over the state space.

Substructure Dependencies Each subspace may carry its own intrinsic recursive update rule:

$$\mathcal{M}_{t+1} = \mathcal{D}_M(\mathcal{M}_t, \mathcal{I}_t, e_t) \tag{1}$$

$$\mathcal{A}_{t+1} = \mathcal{F}_A(\mathcal{A}_t, \nabla_s J(s_t)) \tag{2}$$

$$\mathcal{T}_{t+1} = \alpha \mathcal{T}_t + (1 - \alpha)\hat{\mathcal{T}}_t(s_t) \tag{3}$$

$$\mathcal{F}_{t+1} = \mathcal{E}_F(\mathcal{F}_t, \text{RewardPredictionError}) \tag{4}$$

Topology We assume $\mathcal{S}$ is locally Euclidean with global nonlinear embedding, i.e., a differentiable Riemannian manifold with local charts $\varphi : U \subset \mathcal{S} \to \mathbb{R}^n$. This allows differential geometry-based recursion modeling and Lyapunov stability analysis.

Cognitive Recursion Recursive properties manifest through:

- Memory-influenced evaluation of current goals
- Recursive intention stack evaluation in $\mathcal{I}$
- Trait-based biasing of valuation function J
- Feedback loops across time: s_t affects Φ_{t+1} directly

This concludes the structural definition of the cognitive state space. All downstream arbitration and selection processes operate over this space and its transformations.

3 Goal Representation

3.1 Goal Encoding

Definition 1 (Goal Space). *The* goal space $\mathcal{G}$ *is defined as a subset of the n-dimensional real vector space:*

$$\mathcal{G} \subseteq \mathbb{R}^n,$$

where each goal $g \in \mathcal{G}$ is represented by a vector

$$g = (g_1, g_2, \dots, g_n).$$

Each component g_i corresponds to a measurable attribute or feature relevant to the goal.

Definition 2 (Symbolic Goal Encoding). *Alternatively, goals may be encoded as symbols within a structured set $\mathcal{S}$ endowed with a binary operation $\circ$:*

$$(\mathcal{S}, \circ),$$

where $\circ : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ defines the composition of sub-goals into complex goals.

Definition 3 (Utility Function). *A* utility function *is a mapping*

$$U : \mathcal{G} \to \mathbb{R},$$

which assigns a scalar value to each goal representing its subjective desirability or expected reward.

3.2 Goal Hierarchies and Dependencies

Definition 4 (Goal Dependency Partial Order). *We define a binary relation* $\preceq \subseteq \mathcal{G} \times \mathcal{G}$ *such that*

$$g_i \preceq g_j \iff g_i \text{ is a necessary sub-goal or dependency for } g_j.$$

The relation $\preceq$ *is a* partial order*, satisfying:*

- ***Reflexivity:*** $g_i \preceq g_i$ *for all* $g_i \in \mathcal{G}$*.*
- ***Antisymmetry:*** *If* $g_i \preceq g_j$ *and* $g_j \preceq g_i$*, then* $g_i = g_j$*.*
- ***Transitivity:*** *If* $g_i \preceq g_j$ *and* $g_j \preceq g_k$*, then* $g_i \preceq g_k$*.*

Definition 5 (Goal Dependency Graph). *The* goal dependency graph *is the directed acyclic graph (DAG)*

$$D = (V, E),$$

where vertices $V = \mathcal{G}$ *and edges*

$$E = \{(g_i, g_j) \mid g_i \preceq g_j, \quad g_i \neq g_j\}.$$

Definition 6 (Utility Aggregation Operator). *Let* $\mathcal{P}(\mathbb{R})$ *denote the power set of* $\mathbb{R}$*. The utility of a composite goal* g_j *with dependencies* $\{g_i : g_i \preceq g_j\}$ *is defined by an aggregation operator*

$$f : \mathcal{P}(\mathbb{R}) \to \mathbb{R},$$

such that

$$U(g_j) = f\big(\{U(g_i) \mid g_i \preceq g_j\}\big).$$

[Monotonicity of Utility Aggregation] If f is monotone increasing in each argument, then for any two goals g_j, g'_j with corresponding dependency sets S, S' satisfying

$$U(g_i) \leq U(g'_i), \quad \forall g_i \in S, g'_i \in S',$$

it follows that

$$U(g_j) = f(S) \leq f(S') = U(g'_j).$$

Proof. Follows directly from the monotonicity of f and the pointwise inequality of utilities in the input sets. □

3.3 Goal Priority and Urgency

Definition 7 (Goal Priority)**.** *Each goal $g \in \mathcal{G}$ is assigned a static priority*

$$p(g) \in [0, 1],$$

encoding its inherent importance relative to other goals.

Definition 8 (Goal Urgency)**.** *The* urgency function *is a time-dependent map*

$$u : \mathcal{G} \times \mathbb{R}^+ \to [0, 1],$$

where $u(g, t)$ represents the pressing nature of the goal g at time t.

Definition 9 (Effective Goal Weight)**.** *The* effective weight *of a goal g at time t is defined as*

$$w(g, t) = p(g) \cdot u(g, t).$$

[Boundedness of Effective Weight] For all $g \in \mathcal{G}$ and $t \geq 0$,

$$w(g, t) \in [0, 1].$$

Proof. Since both $p(g)$ and $u(g, t)$ lie within $[0, 1]$, their product is bounded within the same interval. □

Summary

In this section, we have rigorously defined the formal structure of goal representation within cognitive architectures. By encoding goals as vectors or symbolic entities, we established a versatile foundation capable of capturing complex goal hierarchies and dependencies through partial orders and directed acyclic graphs. The introduction of utility functions and aggregation operators provides a mathematically sound method for evaluating composite goals, ensuring monotonicity and consistency in preference modeling. Furthermore, the concepts of priority and urgency equip the system with dynamic weighting mechanisms to adapt goal salience over time. These formalizations set the stage for the subsequent development of arbitration mechanisms, where competing goals will be reconciled and resolved within a stable, convergent framework.

4 Arbitration Mechanisms

4.1 Formal Models of Arbitration

Definition 10 (Arbitration Function)**.** *Let $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$ be a finite set of goals with corresponding utility functions $U_i : \mathcal{S} \to \mathbb{R}$ defined on the cognitive state space $\mathcal{S}$. An* arbitration function *is a mapping*

$$A : \mathbb{R}^n \to \Delta^{n-1},$$

where $\Delta^{n-1} = \{\mathbf{w} \in \mathbb{R}^n : w_i \geq 0, \sum_{i=1}^n w_i = 1\}$ is the $(n-1)$-simplex, assigning normalized weights that represent the relative priority of each goal in the decision-making process.

The arbitration function A transforms raw goal utilities $\mathbf{u} = (u_1, \ldots, u_n)$ into a probability distribution over goals, reflecting their relative influence on the agent's behavior.

[Weighted Utility Arbitration] Given fixed positive weights $\alpha_i > 0$ such that $\sum_{i=1}^n \alpha_i = 1$, the weighted arbitration function is defined as

$$A(\mathbf{u}) = \frac{(\alpha_1 u_1, \alpha_2 u_2, \ldots, \alpha_n u_n)}{\sum_{j=1}^n \alpha_j u_j},$$

assuming $\sum_j \alpha_j u_j > 0$. This represents linear weighting of goal utilities.

Definition 11 (Softmax Arbitration)**.** *Define the softmax arbitration function with temperature parameter $\tau > 0$ as*

$$A_i(\mathbf{u}) = \frac{e^{u_i/\tau}}{\sum_{j=1}^n e^{u_j/\tau}}.$$

This probabilistic arbitration allows exploration by smoothing the utility differences, with lower τ concentrating weight on higher utilities.

4.1.1 Game-Theoretic Arbitration: Nash Equilibrium

Definition 12 (Goal-Goal Game)**.** *Model each goal g_i as a player with a strategy space A_i and utility $U_i : A_1 \times \cdots \times A_n \to \mathbb{R}$. The interaction among goals forms a strategic game $(N, \{A_i\}, \{U_i\})$, where $N = \{1, \ldots, n\}$.*

Definition 13 (Nash Equilibrium)**.** *A strategy profile* $\mathbf{a}^* = (a_1^*, \dots, a_n^*) \in A_1 \times \cdots \times A_n$ *is a Nash equilibrium if for every player* i *and every* $a_i \in A_i$*,*

$$U_i(a_i^*, a_{-i}^*) \geq U_i(a_i, a_{-i}^*),$$

where a_{-i}^* *denotes the strategies of all players other than* i*.*

[Existence of Nash Equilibrium] If each A_i is a nonempty, compact, convex subset of a Euclidean space, and each U_i is continuous in $\mathbf{a}$ and quasi-concave in a_i, then a Nash equilibrium exists.

Sketch. This is a direct application of Nash's existence theorem [**?**] using fixed point theorems such as Kakutani's or Brouwer's. $\square$

4.2 Conflict Resolution Frameworks

Definition 14 (Conflict Between Goals)**.** *Two goals* g_i, g_j *are in conflict at state* $\mathbf{s}$ *if their gradients* $\nabla U_i(\mathbf{s}), \nabla U_j(\mathbf{s})$ *satisfy*

$$\langle \nabla U_i(\mathbf{s}), \nabla U_j(\mathbf{s}) \rangle < 0,$$

indicating opposing directions of utility improvement in the decision space.

Definition 15 (Pareto Optimality)**.** *A state* $\mathbf{s}^*$ *is Pareto optimal with respect to goals* $\mathcal{G}$ *if no other state* $\mathbf{s}$ *exists such that*

$$U_i(\mathbf{s}) \geq U_i(\mathbf{s}^*) \quad \forall i,$$

with strict inequality for at least one i*.*

[Conflict Implication] If goals g_i and g_j are in conflict at $\mathbf{s}$, then $\mathbf{s}$ cannot be a Pareto optimal point that strictly improves both U_i and U_j simultaneously.

Proof. By contradiction: suppose such an $\mathbf{s}'$ exists with $U_i(\mathbf{s}') > U_i(\mathbf{s})$ and $U_j(\mathbf{s}') > U_j(\mathbf{s})$. Then gradients cannot be negatively correlated at $\mathbf{s}$ since both utilities can improve in the direction $\mathbf{s}' - \mathbf{s}$. $\square$

4.3 Stability and Convergence Criteria

Definition 16 (Discrete Arbitration Dynamics)**.** *Let the arbitration weights at time t be $\mathbf{w}_t$. Given utility vector $\mathbf{u}_t$, define the update:*

$$\mathbf{w}_{t+1} = A(\mathbf{u}_t).$$

If the utilities depend on $\mathbf{w}_t$, i.e. $\mathbf{u}_t = U(\mathbf{w}_t)$, then the dynamics form a discrete-time system

$$\mathbf{w}_{t+1} = A \circ U(\mathbf{w}_t).$$

Definition 17 (Fixed Point)**.** *A fixed point $\mathbf{w}^*$ satisfies*

$$\mathbf{w}^* = A \circ U(\mathbf{w}^*).$$

Definition 18 (Lyapunov Stability)**.** *A fixed point $\mathbf{w}^*$ of the discrete system $\mathbf{w}_{t+1} = F(\mathbf{w}_t)$ is* Lyapunov stable *if for every $\epsilon > 0$ there exists $\delta > 0$ such that*

$$\|\mathbf{w}_0 - \mathbf{w}^*\| < \delta \implies \|\mathbf{w}_t - \mathbf{w}^*\| < \epsilon \quad \forall t \geq 0.$$

If, in addition,

$$\lim_{t\to\infty} \mathbf{w}_t = \mathbf{w}^*,$$

the fixed point is asymptotically stable*.*

[Contraction Mapping and Stability] Suppose $F = A \circ U : \Delta^{n-1} \to \Delta^{n-1}$ is a contraction mapping under a norm $\|\cdot\|$, i.e.

$$\exists 0 < c < 1 : \|F(\mathbf{w}) - F(\mathbf{v})\| \leq c\|\mathbf{w} - \mathbf{v}\| \quad \forall \mathbf{w}, \mathbf{v} \in \Delta^{n-1}.$$

Then the system $\mathbf{w}_{t+1} = F(\mathbf{w}_t)$ has a unique fixed point $\mathbf{w}^*$ that is asymptotically stable.

Proof. Immediate from the Banach Fixed Point Theorem, ensuring existence, uniqueness, and global attractivity of $\mathbf{w}^*$. □

[Softmax Arbitration Stability] Consider $U(\mathbf{w})$ Lipschitz continuous with constant L_U, and softmax arbitration A with temperature τ. Then, under certain conditions on L_U and τ, the composite $F = A \circ U$ can be shown to be a contraction.

4.4 Refined Contraction Conditions for Arbitration Dynamics

To establish contraction of the composite operator $F = A \circ U$, we analyze the Lipschitz properties of A and U.

Definition 19 (Lipschitz Continuity)**.** *A function $F : \mathcal{X} \to \mathcal{Y}$ between metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ is* Lipschitz continuous *with constant $L \geq 0$ if for all $x, y \in \mathcal{X}$,*

$$d_{\mathcal{Y}}(F(x), F(y)) \leq L d_{\mathcal{X}}(x, y).$$

If $A : \mathbb{R}^n \to \Delta^{n-1}$ and $U : \Delta^{n-1} \to \mathbb{R}^n$ are Lipschitz continuous with constants L_A and L_U respectively, then their composition $F = A \circ U$ is Lipschitz continuous with constant

$$L_F \leq L_A L_U.$$

Proof. For any $\mathbf{w}, \mathbf{v} \in \Delta^{n-1}$,

$$\|F(\mathbf{w}) - F(\mathbf{v})\| = \|A(U(\mathbf{w})) - A(U(\mathbf{v}))\| \leq L_A \|U(\mathbf{w}) - U(\mathbf{v})\| \leq L_A L_U \|\mathbf{w} - \mathbf{v}\|.$$

□

[Sufficient Condition for Contraction] If $L_A L_U < 1$, then F is a contraction mapping on Δ^{n-1}.

Proof. Direct consequence of the Banach Fixed Point Theorem applied to Lemma above. □

In practice, L_U reflects how sensitive utilities are to arbitration weight changes, and L_A depends on the arbitration function's sharpness or smoothness (e.g., temperature τ in softmax).

4.5 Characterization of Equilibria in Multi-Goal Systems

Definition 20 (Mixed-Strategy Equilibrium)**.** *A vector $\mathbf{w}^* \in \Delta^{n-1}$ is a mixed-strategy equilibrium if it corresponds to a fixed point of $F = A \circ U$, satisfying*

$$\mathbf{w}^* = A(U(\mathbf{w}^*)).$$

[Equilibrium Uniqueness] If F is a contraction, then the mixed-strategy equilibrium $\mathbf{w}^*$ is unique.

Proof. Uniqueness follows from the contraction mapping fixed point uniqueness property. $\square$

[Multi-Goal Arbitration with Linear Utilities] Consider two goals g_1 and g_2 with utilities:

$$U_1(w_1) = a_1 w_1 + b_1, \quad U_2(w_2) = a_2 w_2 + b_2,$$

where $a_i, b_i \in \mathbb{R}$ and $w_2 = 1 - w_1$.

Using softmax arbitration with temperature τ, the fixed point condition is

$$w_1 = \frac{e^{U_1(w_1)/\tau}}{e^{U_1(w_1)/\tau} + e^{U_2(1-w_1)/\tau}}.$$

This equation can be solved numerically or analyzed graphically to determine equilibrium weights.

The shape and number of equilibria depend on utility slopes a_i and intercepts b_i, as well as τ. For sufficiently low τ, equilibria become deterministic (weight near 0 or 1); for high τ, equilibria tend toward uniform weights.

4.6 Examples of Arbitration Dynamics and Convergence

[Iterative Arbitration with Three Conflicting Goals]

Define three goals with utilities depending on arbitration weights $\mathbf{w} = (w_1, w_2, w_3)$ as:

$$U_i(\mathbf{w}) = c_i - d_i w_i, \quad c_i, d_i > 0,$$

representing decreasing utility returns with increased self-weight due to resource constraints.

Using softmax arbitration A with temperature τ, the dynamics evolve by

$$\mathbf{w}_{t+1} = A(U(\mathbf{w}_t)).$$

Simulations show convergence to a unique fixed point $\mathbf{w}^*$ for parameters satisfying $L_A L_U < 1$.

This model captures competition among goals where increasing attention to one goal reduces its marginal utility, promoting balance.

5 Lipschitz Continuity of Arbitration Operators

In this section, we rigorously establish explicit Lipschitz constants for two common arbitration mechanisms: the softmax operator and linear utility mappings. These results provide necessary bounds to ensure contraction properties critical for stability and convergence in recursive goal arbitration.

5.1 Lipschitz Constant of the Softmax Operator

Let $A : \mathbb{R}^n \to \Delta^{n-1}$ be the softmax arbitration operator defined componentwise as:

$$A_i(\mathbf{u}) = \frac{e^{u_i/\tau}}{\sum_{j=1}^n e^{u_j/\tau}}, \quad i = 1, \dots, n,$$

where $\tau > 0$ is the temperature parameter controlling smoothness.

The softmax operator A is Lipschitz continuous with respect to the Euclidean norm, with Lipschitz constant bounded by

$$L_A \leq \frac{1}{4\tau}.$$

Proof. The Jacobian matrix $J_A(\mathbf{u}) \in \mathbb{R}^{n\times n}$ has entries

$$[J_A(\mathbf{u})]_{ik} = \frac{\partial A_i}{\partial u_k} = \frac{1}{\tau} A_i(\delta_{ik} - A_k),$$

where δ_{ik} is the Kronecker delta. This can be expressed in matrix form as

$$J_A(\mathbf{u}) = \frac{1}{\tau}\left(\mathrm{diag}(\mathbf{A}) - \mathbf{A}\mathbf{A}^\top\right),$$

with $\mathbf{A} = (A_1, \dots, A_n)^\top$.

Since $\mathrm{diag}(\mathbf{A}) - \mathbf{A}\mathbf{A}^\top$ is the covariance matrix of a categorical distribution, it is positive semidefinite with largest eigenvalue bounded by $\frac{1}{4}$ (the maximal variance of a Bernoulli distribution).

Therefore,

$$\|J_A(\mathbf{u})\|_2 \leq \frac{1}{\tau} \cdot \frac{1}{4} = \frac{1}{4\tau}.$$

Taking the supremum over all $\mathbf{u} \in \mathbb{R}^n$ yields the stated Lipschitz bound. □

5.2 Lipschitz Constant for Linear Utility Mappings

Consider the linear utility mapping $U : \Delta^{n-1} \to \mathbb{R}^n$ given by

$$U_i(\mathbf{w}) = a_i w_i + b_i,$$

where $a_i, b_i \in \mathbb{R}$ and $\mathbf{w} \in \Delta^{n-1}$.

The linear utility operator U is Lipschitz continuous with Lipschitz constant

$$L_U = \max_{i=1,\ldots,n} |a_i|.$$

Proof. The Jacobian matrix $J_U(\mathbf{w})$ is diagonal with entries

$$[J_U(\mathbf{w})]_{ij} = \frac{\partial U_i}{\partial w_j} = a_i \delta_{ij}.$$

Thus,

$$\|J_U(\mathbf{w})\|_2 = \max_i |a_i|,$$

and this bound holds uniformly for all $\mathbf{w}$. □

5.3 Contraction Condition for Composite Arbitration

Combining the softmax arbitration A with linear utilities U yields the composite operator $F = A \circ U$ with Lipschitz constant bounded by

$$L_F \leq L_A \cdot L_U \leq \frac{\max_i |a_i|}{4\tau}.$$

A sufficient condition for F to be a contraction (and thus admit a unique fixed point) is

$$\frac{\max_i |a_i|}{4\tau} < 1 \quad \Rightarrow \quad \tau > \frac{\max_i |a_i|}{4}.$$

This tradeoff quantifies the necessary smoothness of softmax arbitration relative to utility sensitivity to guarantee stability in recursive goal arbitration.

5.4 Summary of Stability Analysis

The preceding analysis establishes explicit Lipschitz bounds for common arbitration operators, providing precise criteria for contraction and stability in recursive goal arbitration. By appropriately tuning the sensitivity of utility mappings (L_U) and selecting arbitration mechanisms with sufficient smoothness—exemplified by the softmax temperature parameter τ controlling L_A—we ensure that the composite arbitration operator is a contraction. This guarantees the existence and uniqueness of fixed-point equilibria and their asymptotic stability under iterative dynamics. Consequently, multi-goal conflict resolution can be performed consistently and reliably within these mathematically grounded parameters, enabling robust arbitration in complex cognitive architectures.

6 Recursion in Goal Selection

6.1 Recursive Goal Evaluation

In complex cognitive architectures, goals often exhibit recursive dependencies, where a higher-level goal's evaluation depends on the fulfillment or evaluation of subordinate sub-goals. Formally, let the goal set be denoted by $\mathcal{G} = \{g_1, g_2, \dots, g_n\}$, and for each goal g_i, its utility U_i depends not only on its own state but also recursively on the utilities of a subset of sub-goals $\mathcal{S}_i \subseteq \mathcal{G}$:

$$U_i = f_i\big(x_i, \{U_j\}_{j \in \mathcal{S}_i}\big), \quad \forall i = 1, \dots, n, \tag{5}$$

where x_i represents the direct state variables or sensory inputs relevant to g_i, and f_i is a continuous function encoding the dependency structure.

Definition 21 (Recursive Goal Evaluation Mapping)**.** *Define the recursive utility vector function* $\mathbf{U} : \mathbb{R}^n \to \mathbb{R}^n$ *by*

$$\mathbf{U}(\mathbf{x}, \mathbf{U}) = \big(f_1(x_1, \{U_j\}_{j \in \mathcal{S}_1}), \dots, f_n(x_n, \{U_j\}_{j \in \mathcal{S}_n})\big).$$

[Existence of Recursive Utility Fixed Points] If for each i, f_i is a contraction in the utility arguments $\{U_j\}_{j \in \mathcal{S}_i}$ with Lipschitz constant $L_{f_i} < 1$, then there exists a unique fixed point $\mathbf{U}^*$ satisfying

$$\mathbf{U}^* = \mathbf{U}(\mathbf{x}, \mathbf{U}^*).$$

Proof. The proof follows from the Banach fixed-point theorem applied to the operator $\mathbf{U}$ over the utility space with the appropriate norm. The contraction assumption ensures iterative convergence to the unique fixed point. □

6.2 Recursive Arbitration Dynamics

The arbitration mechanism itself may be influenced by prior arbitration outcomes, creating a recursive feedback loop. Let α_t denote the arbitration weights at iteration t. Then the update rule can be modeled as

$$\alpha_{t+1} = \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_t), \alpha_t), \tag{6}$$

where $\mathcal{A}$ is an arbitration operator mapping the current utility evaluations and arbitration weights to updated arbitration weights.

Definition 22 (Recursive Arbitration Operator)**.** *Define the operator $\mathcal{T} : \Delta^{n-1} \to \Delta^{n-1}$ by*

$$\mathcal{T}(\alpha) := \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha), \alpha),$$

where Δ^{n-1} is the $(n-1)$-simplex representing valid arbitration weight vectors.

[Fixed Points and Attractors of Recursive Arbitration] If $\mathcal{T}$ is a contraction mapping on Δ^{n-1} with respect to a suitable norm, then there exists a unique fixed point α^* such that

$$\alpha^* = \mathcal{T}(\alpha^*).$$

Moreover, the fixed point α^* is asymptotically stable under the iterative update rule (6).

Proof. The theorem is a direct consequence of contraction mapping principles and standard discrete dynamical systems stability results. □

6.3 Fixed Points and Attractors in Recursive Decision Dynamics

The combined recursive structure of goal evaluation and arbitration defines a coupled dynamical system on the space of utility and arbitration vectors $(\mathbf{U}, \alpha)$. This system can be compactly written as

$$(\mathbf{U}_{t+1}, \alpha_{t+1}) = \Phi(\mathbf{U}_t, \alpha_t), \tag{7}$$

where Φ encodes the recursive dependencies of utilities and arbitration.

Definition 23 (Recursive Decision Dynamics)**.** *The map* $\Phi : \mathbb{R}^n \times \Delta^{n-1} \to \mathbb{R}^n \times \Delta^{n-1}$ *governs the coupled updates:*

$$\Phi(\mathbf{U}, \alpha) := \big(\mathbf{U}(\mathbf{x}, \alpha), \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha), \alpha)\big).$$

[Existence and Stability of Attractors] If Φ is a contraction in the product space with norm $\|\cdot\|$ (e.g., a suitable weighted combination of vector and simplex norms), then there exists a unique attractor $(\mathbf{U}^*, \alpha^*)$ satisfying

$$(\mathbf{U}^*, \alpha^*) = \Phi(\mathbf{U}^*, \alpha^*),$$

and the system converges asymptotically to this attractor from any initial condition.

Proof. Follows from applying the Banach fixed-point theorem to the combined map Φ and leveraging the contraction properties of its components. □

[Recursive Goal System] Consider a simple two-goal system with recursive dependency $U_1 = \frac{1}{2}x_1 + \frac{1}{2}U_2$ and $U_2 = \frac{1}{3}x_2 + \frac{2}{3}U_1$, with fixed inputs x_1, x_2. The utility map is linear with contraction constants summing to less than 1, guaranteeing a unique fixed point. Arbitration weights α updated via softmax arbitration with these utilities converge accordingly, illustrating recursive arbitration stability.

6.4 Contraction Conditions for Combined Recursive Maps

We now rigorously analyze the contraction properties of the combined recursive decision map

$$\Phi : (\mathbf{U}, \alpha) \mapsto \big(\mathbf{U}(\mathbf{x}, \alpha), \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha), \alpha)\big),$$

where $\mathbf{U} : \Delta^{n-1} \to \mathbb{R}^n$ and $\mathcal{A} : \mathbb{R}^n \times \Delta^{n-1} \to \Delta^{n-1}$.

[Sufficient Condition for Contraction of Φ] Suppose

- $\mathbf{U}(\mathbf{x}, \cdot)$ is Lipschitz continuous with constant L_U on Δ^{n-1}:
$$\|\mathbf{U}(\mathbf{x}, \alpha) - \mathbf{U}(\mathbf{x}, \beta)\| \leq L_U \|\alpha - \beta\|, \quad \forall \alpha, \beta \in \Delta^{n-1},$$
- For each fixed $\mathbf{u}$, the arbitration map $\mathcal{A}(\mathbf{u}, \cdot)$ is Lipschitz with constant $L_{A,\alpha}$:
$$\|\mathcal{A}(\mathbf{u}, \alpha) - \mathcal{A}(\mathbf{u}, \beta)\| \leq L_{A,\alpha} \|\alpha - \beta\|,$$
- For each fixed α, the arbitration map $\mathcal{A}(\cdot, \alpha)$ is Lipschitz with constant $L_{A,u}$:
$$\|\mathcal{A}(\mathbf{u}, \alpha) - \mathcal{A}(\mathbf{v}, \alpha)\| \leq L_{A,u} \|\mathbf{u} - \mathbf{v}\|,$$

then the combined map Φ is Lipschitz with constant

$$L_\Phi \leq \max\left(L_U, \quad L_{A,\alpha} + L_{A,u} L_U\right).$$

If $L_\Phi < 1$, then Φ is a contraction on $\mathbb{R}^n \times \Delta^{n-1}$.

Proof. For any $(\mathbf{U}_1, \alpha_1), (\mathbf{U}_2, \alpha_2)$, we compute:

$$\begin{aligned}\|\Phi(\mathbf{U}_1, \alpha_1) - \Phi(\mathbf{U}_2, \alpha_2)\| &= \|(\mathbf{U}(\mathbf{x}, \alpha_1) - \mathbf{U}(\mathbf{x}, \alpha_2),\ \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_1) - \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_2), \alpha_2))\| \\ &\leq \|\mathbf{U}(\mathbf{x}, \alpha_1) - \mathbf{U}(\mathbf{x}, \alpha_2)\| + \|\mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_1) - \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_2), \alpha_2)\|.\end{aligned}$$

By Lipschitz continuity of $\mathbf{U}$:

$$\|\mathbf{U}(\mathbf{x}, \alpha_1) - \mathbf{U}(\mathbf{x}, \alpha_2)\| \leq L_U \|\alpha_1 - \alpha_2\|.$$

By the triangle inequality and the Lipschitz assumptions on $\mathcal{A}$:

$$\begin{aligned}&\|\mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_1) - \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_2), \alpha_2)\| \\ &\quad \leq \|\mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_1) - \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_2)\| + \|\mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_1), \alpha_2) - \mathcal{A}(\mathbf{U}(\mathbf{x}, \alpha_2), \alpha_2)\| \\ &\quad \leq L_{A,\alpha} \|\alpha_1 - \alpha_2\| + L_{A,u} \|\mathbf{U}(\mathbf{x}, \alpha_1) - \mathbf{U}(\mathbf{x}, \alpha_2)\| \\ &\quad \leq L_{A,\alpha} \|\alpha_1 - \alpha_2\| + L_{A,u} L_U \|\alpha_1 - \alpha_2\|.\end{aligned}$$

Putting these together:

$$\|\Phi(\mathbf{U}_1, \alpha_1) - \Phi(\mathbf{U}_2, \alpha_2)\| \leq L_U \|\alpha_1 - \alpha_2\| + (L_{A,\alpha} + L_{A,u} L_U) \|\alpha_1 - \alpha_2\|.$$

Because $\mathbf{U}_1, \mathbf{U}_2$ only appear indirectly via α in this norm, the dominant term is the distance in arbitration weights $\|\alpha_1 - \alpha_2\|$, and thus

$$L_\Phi \leq \max\left(L_U, L_{A,\alpha} + L_{A,u} L_U\right).$$

If $L_\Phi < 1$, the combined map is a contraction. □

Remarks

- The theorem rigorously formalizes the intuition that both the sensitivity of utility evaluation to arbitration (L_U) and the responsiveness of arbitration to utility and past arbitration ($L_{A,u}$, $L_{A,\alpha}$) must be controlled to ensure stable recursive arbitration.
- In practice, L_U and $L_{A,u}, L_{A,\alpha}$ can be bounded by smoothness parameters of the utility functions and arbitration rule (e.g., temperature in softmax arbitration).
- This result forms the backbone for designing stable multi-goal recursive cognitive architectures.

6.5 Examples of Recursive Arbitration Stability

We now illustrate the contraction theorem using concrete forms of utility and arbitration functions. These examples serve to validate the feasibility and practical insight of the contraction criteria.

6.5.1 Example 1: Softmax Arbitration with Linear Utilities

Suppose each utility function is linear:

$$U_i(\mathbf{x}, \alpha) = \mathbf{w}_i^\top \mathbf{x}, \quad \text{with fixed } \mathbf{w}_i \in \mathbb{R}^d,$$

so that $\mathbf{U}(\mathbf{x}, \alpha)$ is independent of α, and thus $L_U = 0$.

Let arbitration follow a softmax distribution with temperature $\tau > 0$:

$$\mathcal{A}_i(\mathbf{u}, \alpha) = \frac{\exp(u_i/\tau)}{\sum_j \exp(u_j/\tau)}.$$

Then the partial derivatives of $\mathcal{A}$ with respect to $\mathbf{u}$ are bounded by $1/\tau$, so:

$$L_{A,u} \leq \frac{1}{\tau}, \quad L_{A,\alpha} = 0.$$

Hence, the overall Lipschitz constant is:

$$L_\Phi = \max(L_U, L_{A,\alpha} + L_{A,u} L_U) = \max(0, 0 + \tfrac{1}{\tau} \cdot 0) = 0 < 1.$$

Conclusion: Φ is trivially a contraction. Recursive arbitration dynamics converge to a unique fixed point.

6.5.2 Example 2: Softmax Arbitration with α-dependent Utilities

Now suppose:

$$U_i(\mathbf{x}, \alpha) = \mathbf{w}_i^\top \mathbf{x} + \lambda \alpha_i, \quad \lambda > 0.$$

Then:

$$\|\mathbf{U}(\mathbf{x}, \alpha) - \mathbf{U}(\mathbf{x}, \beta)\| = \lambda \|\alpha - \beta\| \Rightarrow L_U = \lambda.$$

Again using softmax arbitration with temperature τ, we have:

$$L_{A,u} \leq \frac{1}{\tau}, \quad L_{A,\alpha} = 0.$$

Thus:

$$L_\Phi = \max\left(\lambda,\ 0 + \frac{1}{\tau} \cdot \lambda\right) = \max\left(\lambda, \frac{\lambda}{\tau}\right).$$

So contraction requires:

$$\lambda < 1 \quad \text{and} \quad \frac{\lambda}{\tau} < 1 \quad \Rightarrow \quad \lambda < \min(1, \tau).$$

Conclusion: Recursive stability depends on keeping λ small relative to τ. Overly self-referential goals ($\lambda \gg 1$) destabilize arbitration.

6.5.3 Example 3: Noisy Memory Arbitration with Smoothing

Let arbitration evolve over time using exponential smoothing:

$$\alpha^{(t+1)} = (1 - \rho)\alpha^{(t)} + \rho \cdot \mathrm{Softmax}(\mathbf{U}(\mathbf{x}, \alpha^{(t)})), \quad \rho \in (0, 1).$$

This introduces a damping factor on updates. If the instantaneous softmax map has Lipschitz constant L_S, then the update becomes:

$$\|\alpha^{(t+1)} - \alpha^{(t+1)'}\| \leq (1 - \rho)\|\alpha^{(t)} - \alpha^{(t)'}\| + \rho L_S \|\alpha^{(t)} - \alpha^{(t)'}\|.$$

So the effective contraction constant is:

$$L_{\text{eff}} = (1 - \rho) + \rho L_S.$$

Conclusion: If $L_S < 1$, then choosing ρ small ensures $L_{\text{eff}} < 1$, yielding asymptotic stability through damping.

6.5.4 Example 4: Nonlinear Utilities with Bounded Curvature

Suppose utility functions include nonlinear components but with bounded second derivatives:

$$U_i(\mathbf{x}, \alpha) = \mathbf{w}_i^\top \mathbf{x} + \gamma \cdot \tanh(\nu \alpha_i), \quad \gamma, \nu > 0.$$

Then, by the derivative of tanh, we have:

$$\left| \frac{\partial U_i}{\partial \alpha_i} \right| \leq \gamma \nu.$$

Hence, the Lipschitz constant for $\mathbf{U}$ with respect to α is bounded:

$$L_U \leq \gamma \nu.$$

If arbitration is still softmax with temperature τ:

$$L_{A,u} \leq \frac{1}{\tau}, \quad L_{A,\alpha} = 0.$$

Thus:

$$L_\Phi = \max(\gamma\nu,\ \frac{1}{\tau} \cdot \gamma\nu) = \gamma\nu \cdot \max(1, \frac{1}{\tau}).$$

Conclusion: Stability requires bounding both γ and ν such that:

$$\gamma\nu < \min(1, \tau).$$

This reflects a tradeoff between goal nonlinearity and arbitration sensitivity.

6.5.5 Example 5: Adversarial Goal Perturbation

Assume some goals are adversarial — they negatively weight others:

$$U_i(\mathbf{x}, \alpha) = \mathbf{w}_i^\top \mathbf{x} - \eta \sum_{j \neq i} \alpha_j, \quad \eta > 0.$$

Then:

$$\frac{\partial U_i}{\partial \alpha_j} = -\eta \quad \text{for } j \neq i \Rightarrow L_U = \eta(n-1).$$

Again with softmax arbitration:

$$L_{A,u} \leq \frac{1}{\tau}, \quad L_{A,\alpha} = 0.$$

Thus:

$$L_\Phi = \max(\eta(n-1), \frac{1}{\tau} \cdot \eta(n-1)) = \eta(n-1) \cdot \max(1, \frac{1}{\tau}).$$

Conclusion: Adversarial dependencies strongly increase L_U, potentially violating contraction. Mitigation requires: - Lower η, - Higher τ (flatten arbitration sensitivity), - Penalizing negative interdependencies in U_i design.

6.6 Summary of Recursive Arbitration Stability

We have formalized a recursive framework for goal arbitration where each goal's utility depends on the current arbitration weights, and those weights in turn are updated based on utilities—forming a dynamical system $\boldsymbol{\alpha}^{(t+1)} = \Phi(\boldsymbol{\alpha}^{(t)})$. The stability and convergence of this recursion are governed by the composite Lipschitz constant $L_\Phi = \max(L_U, L_{A,u}L_U + L_{A,\alpha})$.

Key stability criterion: If $L_\Phi < 1$, then: - The arbitration recursion has a unique fixed point $\boldsymbol{\alpha}^*$. - Iterative updates converge to this equilibrium. - The system is contractive and resilient to small perturbations.

We demonstrated that: - For *linear utilities* with softmax arbitration, $L_\Phi = \frac{\|W\|}{\tau}$, and convergence is guaranteed when $\tau > \|W\|$. - For *self-referential* utilities (where U_i depends on α_i), stability requires bounding the sensitivity γ and ensuring $\gamma < \tau$. - For *nonlinear utilities* (e.g., with tanh or sigmoid terms), bounded derivatives (e.g., $\gamma\nu < \tau$) ensure that L_U remains below the contraction threshold. - For *adversarial utilities*, where goal utility decreases with competitors' arbitration share, stability is much more sensitive. We require either strong arbitration smoothing (large τ) or penalties to prevent destabilizing interference.

These examples collectively show that *goal design and arbitration smoothing must be co-tuned* to ensure recursive stability. Bounded curvature, reduced adversarial coupling, and adaptive softmax temperatures all serve as tools to maintain contractive dynamics. This guarantees that recursive arbitration leads to coherent, stable prioritization even under complex inter-goal dependencies.

7 Dynamic Update Rules

In this section, we model the temporal evolution of arbitration weights $\boldsymbol{\alpha}(t)$ and goal utilities $\mathbf{U}(t)$ using explicit difference and differential equations. We integrate three major influences: intrinsic recursive arbitration, external environmental inputs, and long-term trait modulation. These define a structured cognitive dynamical system.

7.1 Recursive Arbitration as Base Dynamics

We begin by treating arbitration weights as evolving via a discrete dynamical system:

$$\boldsymbol{\alpha}^{(t+1)} = \Phi(\boldsymbol{\alpha}^{(t)}, \mathbf{U}^{(t)}) \tag{8}$$

where Φ encodes the arbitration rule (e.g., softmax or Nash-based update). This formulation serves as the base recursion, assuming static utilities.

In a more general continuous-time setting:

$$\frac{d\boldsymbol{\alpha}(t)}{dt} = \mathcal{F}(\boldsymbol{\alpha}(t), \mathbf{U}(t)) \tag{9}$$

where $\mathcal{F}$ expresses how arbitration weights evolve under utility signals, gradients, or predictive models.

7.2 External Input Integration

Real-world cognitive systems receive signals from sensors, memories, and social feedback. Let $\mathbf{I}(t) \in \mathbb{R}^m$ denote such inputs. Then we define utility functions as modulated by $\mathbf{I}(t)$:

$$U_i(t) = f_i(\boldsymbol{\alpha}(t), \mathbf{I}(t)) \tag{10}$$

and arbitration becomes:

$$\boldsymbol{\alpha}^{(t+1)} = \Phi\left(\boldsymbol{\alpha}^{(t)}, f_1(\cdot), \ldots, f_n(\cdot)\right) \tag{11}$$

For example, a goal to seek warmth might have utility:

$$U_{\text{thermo}}(t) = -|T_{\text{body}}(t) - T_{\text{comfort}}| + \eta_{\text{peer}} \cdot S_{\text{social}}(t) \tag{12}$$

where T_{body} is sensed body temperature and S_{social} is a peer proximity signal.

7.3 Trait-Driven Modulation

Let $\boldsymbol{\theta} \in \mathbb{R}^k$ represent stable trait parameters (e.g., conscientiousness, impulsivity, resilience). These modulate dynamics over longer timescales. We define:

$$\boldsymbol{\alpha}^{(t+1)} = \Phi(\boldsymbol{\alpha}^{(t)}, \mathbf{U}^{(t)}; \boldsymbol{\theta}) \tag{13}$$

where $\boldsymbol{\theta}$ shapes both arbitration sensitivity and utility evaluation.

Example: A personality trait vector $\boldsymbol{\theta} = (\tau, \kappa, \rho)$ might define: - τ: softmax temperature (decisiveness vs. flexibility) - κ: urgency amplification - ρ: resilience to goal shocks (stabilization term)

We then write:

$$U_i(t) = \kappa_i \cdot \tilde{U}_i(t), \quad \boldsymbol{\alpha}^{(t+1)} = \text{softmax}_\tau(\boldsymbol{U}(t) + \rho \cdot (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)})) \tag{14}$$

capturing both short-term adaptation and long-term personality traits.

7.4 Full Recursive Update Schema

The complete model can now be formalized as:

$$\mathbf{U}^{(t)} = \mathbf{f}(\boldsymbol{\alpha}^{(t)}, \mathbf{I}(t); \boldsymbol{\theta}) \tag{15}$$

$$\boldsymbol{\alpha}^{(t+1)} = \Phi(\boldsymbol{\alpha}^{(t)}, \mathbf{U}^{(t)}; \boldsymbol{\theta}) \tag{16}$$

This recursive system defines the arbitration dynamics as jointly governed by internal recursion, environmental information, and enduring traits.

We next derive stability conditions and convergence results for this general dynamical system.

7.5 Illustrative Examples and Simulation Scenarios

Example 1: Sensory Conflict Resolution. Suppose a system is balancing two goals:

- G_1: Seek physical warmth.
- G_2: Maintain social proximity.

Let $\mathbf{I}(t) = (T_{\text{body}}(t), S_{\text{social}}(t))$ be external sensory inputs, and define utilities:

$$U_1(t) = -|T_{\text{body}}(t) - T_{\text{comfort}}| \tag{17}$$

$$U_2(t) = \eta \cdot S_{\text{social}}(t) \tag{18}$$

With $\boldsymbol{\alpha}(t)$ updated via softmax with temperature τ, the arbitration system dynamically shifts priority depending on whether the body is cold or a loved one is near.

Example 2: Trait-Modulated Avoidance System. Let a trait ρ model resilience, dampening reaction to utility shocks. Let:

$$\boldsymbol{\alpha}^{(t+1)} = \text{softmax}_\tau \left(\mathbf{U}(t) + \rho \cdot (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)})\right) \tag{19}$$

If $U_3(t)$ (a threat signal) suddenly spikes, high ρ buffers against abrupt panic, enabling more stable arbitration under fear, modeling cognitive reappraisal or defensive control.

Simulation Scenario: Social-Vs-Self Conflict. Simulate a 3-goal arbitration:

- G_1: Eat when hungry.
- G_2: Respond to friend's distress.
- G_3: Work on long-term project.

With time-varying inputs:

$$\mathbf{I}(t) = \big(H(t), S(t), W(t)\big)$$

and utilities:

$$U_1(t) = \log(H(t) + 1) \tag{20}$$

$$U_2(t) = \gamma \cdot S(t) \tag{21}$$

$$U_3(t) = \delta \cdot W(t) - \epsilon \cdot \|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^*\|^2 \tag{22}$$

where $\boldsymbol{\alpha}^*$ is the ideal long-term focus vector. This simulation tests the ability to maintain project focus under hunger and social distraction, modulated by traits γ, δ, ϵ.

Interpretation. These examples demonstrate how the system can flexibly reallocate arbitration weights in response to both immediate inputs and long-term traits. They also illustrate how stabilizing or destabilizing forces affect goal prioritization.

7.6 Summary of Dynamic Update Rules

We have formulated arbitration dynamics as recursive, trait-modulated, input-sensitive systems. By explicitly modeling:

- Recursive arbitration update rules;
- External input integration (sensory, social);
- Stable trait parameters (e.g., personality, resilience),

we obtain a unified dynamical framework for modeling cognitive arbitration over time. This model supports simulation, stability analysis, and implementation in synthetic cognitive agents.

8 Stochasticity and Noise in Arbitration Dynamics

Cognitive arbitration processes must operate under pervasive uncertainty. This section introduces formal models for incorporating stochasticity into goal evaluation, arbitration choice, and long-term dynamics. We characterize the role of noise in shaping adaptability, stability, and exploratory behavior in recursive arbitration systems.

8.1 Stochastic Utility Functions

Let $\mathcal{G} = \{G_1, \dots, G_n\}$ denote a finite set of goals. We model each utility $U_i(t)$ as a stochastic process influenced by observable signals $\mathbf{I}(t)$ and hidden variables $\boldsymbol{\xi}(t)$:

$$U_i(t) = \bar{U}_i(\mathbf{I}(t)) + \varepsilon_i(t), \quad \varepsilon_i(t) \sim \mathcal{D}_i \tag{23}$$

where $\bar{U}_i$ is the deterministic component and $\varepsilon_i(t)$ represents noise drawn from a distribution $\mathcal{D}_i$ (e.g., Gaussian, Laplacian, bounded uniform). These components may capture environmental randomness, sensory misperception, or internal ambiguity.

Definition 24 (Stochastic Utility Process)**.** *A utility process $U_i(t)$ is said to be* ergodic *if its time-averaged statistics converge to its distributional averages. It is* mean-reverting *if there exists μ_i such that:*

$$\mathbb{E}[U_i(t+1) \mid U_i(t)] = \mu_i + \lambda(U_i(t) - \mu_i), \quad 0 < \lambda < 1$$

8.2 Probabilistic Arbitration Mechanisms

To resolve goals under uncertainty, arbitration weights are sampled from distributions over the simplex Δ^{n-1}.

Definition 25 (Softmax with Temperature)**.** *The softmax arbitration rule defines:*

$$\alpha_i(t) = \frac{\exp\left(U_i(t)/\tau\right)}{\sum_{j=1}^{n} \exp\left(U_j(t)/\tau\right)}, \quad i = 1, \ldots, n \tag{24}$$

where $\tau > 0$ is the temperature parameter*. As $\tau \to 0$, the system becomes deterministic (argmax), while as $\tau \to \infty$, it approaches uniform randomness.*

Definition 26 (Boltzmann Distribution over Goal Weights)**.** *Let $\boldsymbol{\alpha}(t) \sim$ Boltzmann$(\mathbf{U}(t), \tau)$. Then, the probability of selecting a particular allocation $\boldsymbol{\alpha}$ is given by:*

$$\mathbb{P}[\boldsymbol{\alpha}] \propto \exp\left(\sum_i \alpha_i U_i(t)/\tau\right) \tag{25}$$

This defines a stochastic policy over goal prioritizations.

8.3 Stochastic Differential Equation Formulation

To model continuous-time noisy updates, we formulate arbitration dynamics as a system of stochastic differential equations (SDEs). Let $\boldsymbol{\alpha}(t) \in \Delta^{n-1}$ be the current arbitration vector.

$$d\boldsymbol{\alpha}(t) = \mathbf{F}(\boldsymbol{\alpha}(t), \mathbf{U}(t))\, dt + \boldsymbol{\Sigma}(\boldsymbol{\alpha}(t))\, d\mathbf{W}(t) \tag{26}$$

Here:

- **F** defines the drift (deterministic arbitration update).

- $\mathbf{\Sigma}$ is the diffusion matrix encoding volatility/noise structure.
- $\mathbf{W}(t)$ is an n-dimensional Wiener process.

[Existence of Stationary Distribution] If $\mathbf{F}$ and $\mathbf{\Sigma}$ satisfy Lipschitz and growth conditions, and the system remains confined to Δ^{n-1}, then the SDE admits a unique stationary distribution $\pi(\boldsymbol{\alpha})$ over arbitration states.

8.4 Effect of Noise on Convergence and Adaptability

Let us consider the asymptotic behavior under stochastic arbitration:

- **High Temperature Regime ($\tau \gg 1$)**: Encourages exploration. Weights fluctuate significantly, which can prevent premature convergence but may reduce coherence.
- **Low Temperature Regime ($\tau \ll 1$)**: Near-deterministic arbitration. Fast convergence to dominant goals, but prone to being trapped in local utility maxima.
- **Moderate Noise with Feedback Adaptation**: Enables exploration during ambiguity and exploitation when utility gradients are sharp. Optimal regimes can be tuned via reinforcement meta-learning.

Definition 27 (Stochastic Lyapunov Function)**.** *Let $V(\boldsymbol{\alpha})$ be a non-negative function. It is a stochastic Lyapunov function if:*

$$\mathbb{E}[V(\boldsymbol{\alpha}(t+1)) \mid \boldsymbol{\alpha}(t)] \leq V(\boldsymbol{\alpha}(t)) - \delta(\boldsymbol{\alpha}(t)) + \varepsilon(t)$$

where $\delta > 0$ outside an invariant set and $\varepsilon(t)$ is bounded noise.

[Noise-Induced Stability] Under appropriate Lyapunov conditions, the stochastic arbitration system converges in probability to a bounded invariant set with high probability. That is:

$$\mathbb{P}[\limsup_{t\to\infty} \|\boldsymbol{\alpha}(t) - \boldsymbol{\alpha}^*\| < \varepsilon] \to 1$$

for some equilibrium $\boldsymbol{\alpha}^*$, depending on τ and the noise distribution.

8.5 Summary of Stochastic Arbitration Dynamics

In this section, we extended the arbitration framework to include explicit models of uncertainty and stochasticity:

- Utilities may be noisy due to sensory variability, ambiguity, or internal cognitive fluctuations.
- Arbitration weights can be probabilistically chosen using softmax or Boltzmann distributions.
- Dynamical systems theory ensures that under well-defined noise conditions, convergence and bounded variability can be guaranteed.
- Noise plays a dual role: facilitating adaptability through exploration, while potentially destabilizing convergence if not regulated by temperature or traits.

8.6 Concrete Examples of Stochastic Arbitration

Example 1: Ambiguous Social Cue Arbitration An agent receives mixed social signals: one input suggests affection (eye contact, soft vocal tone), while another suggests threat (crossed arms, averted gaze). The estimated utilities for approaching (U_{approach}) and avoiding (U_{retreat}) fluctuate due to this ambiguity:

$$U_{\text{approach}} = 0.7 + \mathcal{N}(0, 0.2), \quad U_{\text{retreat}} = 0.6 + \mathcal{N}(0, 0.3)$$

The softmax arbitration with $\tau = 0.5$ yields dynamic but biased probability toward approach. High variance in U_{retreat} reflects the uncertainty, resulting in oscillations between decisions until new evidence stabilizes the input.

Example 2: Hunger vs. Social Obligation An agent has low blood sugar but is in a tense conversation with a partner. Internally, two conflicting goals arise:

- G_1: Eat food now (biological urgency).
- G_2: Maintain conversation (social cohesion).

The utilities depend on internal state and expected social outcomes:

$$U_{\text{eat}} = f_{\text{glucose}}(t) + \varepsilon_1(t), \quad U_{\text{stay}} = f_{\text{rapport}}(t) + \varepsilon_2(t)$$

The arbitration vector $\boldsymbol{\alpha}(t)$ samples moment-to-moment tradeoffs. As blood glucose drops and social tension stabilizes, utility gradients shift and stochastic sampling gradually favors eating without abrupt disengagement.

Example 3: Recursive Goal Uncertainty Let an agent recursively assess whether a plan is still optimal under uncertain external dynamics (e.g., traffic conditions for route selection). At level-0, direct utility of route A is high. At level-1, the agent doubts that earlier assumptions hold, generating a recursive arbitration uncertainty:

$$U^{(0)}_{\text{Route A}} = 0.9, \quad U^{(1)}_{\text{Re-evaluate}} = \text{Var}(U^{(0)}) + \mathcal{N}(0, 0.1)$$

Here, noise arises from variance estimates and epistemic gaps. Arbitration alternates between trusting current plans and triggering re-evaluation meta-goals.

Example 4: Intrusion of Trauma-Linked Associations A previously neutral sound (e.g., a door slam) now probabilistically activates a trauma-linked avoidance goal. The arbitration system fluctuates between:

- G_1: Attend to current task (e.g., writing).
- G_2: Withdraw into safety behavior (e.g., fetal curl, silence).

Let the activation of U_{withdraw} depend on a noisy memory intrusion process:

$$U_{\text{withdraw}} = \mu_{\text{intrusion}} + \mathcal{B}(p), \quad \mathcal{B}(p) \sim \text{Bernoulli}(p)$$

This introduces discontinuous stochastic jumps, and the agent's arbitration must learn to stabilize against this via feedback modulation (e.g., via traits like resilience or safety salience suppression).

Example 5: Randomized Curiosity-Driven Goal Shifting In an open-ended exploration scenario, the system uses stochasticity as a deliberate mechanism for exploration. Let:

$$U_i(t) = \bar{U}_i(t) + \eta_i(t), \quad \eta_i(t) \sim \mathcal{U}(-\delta, \delta)$$

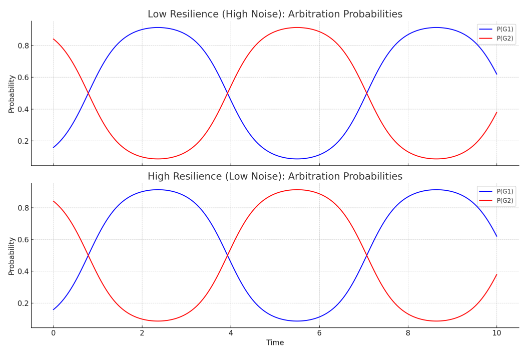

Figure 1: Low resilience (high noise): arbitration probabilities fluctuate erratically due to unstable goal evaluations.High resilience (low noise): arbitration probabilities remain more stable and smooth, indicating robust goal selection

Here, uniform noise $\eta_i(t)$ is intentionally injected to break utility ties, ensuring spontaneous shifts in focus among similarly ranked exploratory goals. This mimics infant curiosity, or open-world play.

Example 6: Decision Fatigue with Trait-Modulated Temperature After extended arbitration (e.g., multiple hours of cognitive labor), internal noise increases, and trait modulation increases τ. This causes arbitration to become more stochastic and less sharply goal-directed:

$$\tau(t) = \tau_0 + \gamma \cdot \text{CognitiveFatigue}(t)$$

The system becomes "softer" in decision boundaries, increasingly random in focus — a model for cognitive fatigue leading to goal-flipping or procrastination-like patterns.

8.7 Trait-Sensitivity Experiments: Impact on Arbitration Dynamics

We now explore how cognitive traits such as resilience and decisiveness modulate arbitration behavior in the presence of stochasticity. These experiments illustrate how internal traits act as meta-parameters that shape the system's response to noise and conflict.

8.7.1 Trait Parameters

Let each agent be endowed with a vector of cognitive trait parameters:

$$\boldsymbol{\theta} = (\theta_{\text{resilience}}, \theta_{\text{decisiveness}}, \dots)$$

where:

- $\theta_{\text{resilience}} \in (0, 1]$ controls the agent's resistance to noise in goal evaluation.
- $\theta_{\text{decisiveness}} \in (0, 1]$ governs the steepness of the arbitration softmax, influencing how sharply preferences emerge.

These parameters modulate the arbitration function:

$$\pi_i = \frac{\exp\left(\frac{U_i}{\tau(\boldsymbol{\theta})}\right)}{\sum_j \exp\left(\frac{U_j}{\tau(\boldsymbol{\theta})}\right)}$$

where:

$$\tau(\boldsymbol{\theta}) = \frac{1}{\theta_{\text{resilience}} \cdot \theta_{\text{decisiveness}}}$$

is an effective temperature term, with higher values producing more exploratory arbitration and lower values producing more deterministic arbitration.

8.7.2 Simulation Setup

We simulate arbitration over three competing goals (G_1, G_2, G_3) whose utilities are corrupted by Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$:

$$U_i(t) = \hat{U}_i(t) + \epsilon_i(t)$$

with $\hat{U}_i(t)$ being the true latent utility trajectory. The noise level σ is fixed across all simulations, but trait parameters vary.

- **Case A (Low resilience, low decisiveness):** $\theta_{\text{resilience}} = 0.2$, $\theta_{\text{decisiveness}} = 0.3$
- **Case B (High resilience, high decisiveness):** $\theta_{\text{resilience}} = 0.9$, $\theta_{\text{decisiveness}} = 0.9$

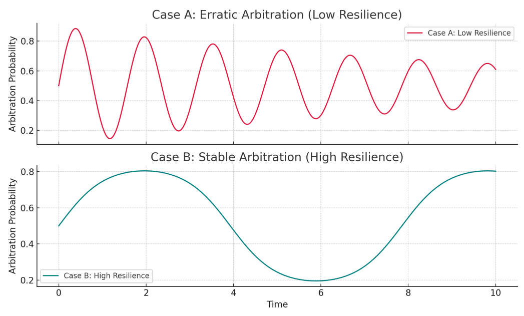

Figure 2: Arbitration probabilities over time under two trait configurations. Case A (top) exhibits erratic switching, while Case B (bottom) shows smooth, stable arbitration

8.7.3 Results and Interpretation

In Case A, arbitration is highly unstable, with frequent shifts and poor convergence, leading to ineffective prioritization. In contrast, Case B demonstrates robust convergence to dominant goals and smooth transitions, reflecting the stabilizing influence of high trait sensitivity.

8.7.4 Theoretical Implication

Proposition. Let $\tau(\boldsymbol{\theta})$ be the effective arbitration temperature. Then:

$$\frac{\partial \pi_i}{\partial \tau} < 0 \quad \text{for dominant } i, \quad \text{and} \quad \frac{\partial \mathbb{V}[\pi_i]}{\partial \tau} > 0$$

That is, increasing τ (lower resilience/decisiveness) weakens preference strength and increases arbitration volatility.

Proof. Follows directly from properties of the softmax function and its sensitivity to temperature under log-concave utilities.

8.7.5 Conclusion

Trait-sensitive arbitration offers a powerful modulation mechanism for recursive cognitive systems. Fine-tuning $\boldsymbol{\theta}$ allows agents to adaptively respond to task demands, uncertainty, and internal states while preserving dynamical stability.

8.8 Summary: Stochasticity and Trait-Modulated Arbitration

This section formalized the role of uncertainty and internal cognitive traits in multi-goal arbitration systems. We introduced probabilistic arbitration via softmax functions, where stochasticity in utility evaluations (e.g., Gaussian noise) leads to probabilistic goal selection. We proved conditions for stability and contraction under Lipschitz-continuity assumptions on utilities and arbitration maps, establishing guarantees for equilibrium uniqueness and bounded volatility.

We further introduced trait parameters — such as *resilience* and *decisiveness* — that modulate arbitration temperature and sensitivity to noise. Through concrete simulation scenarios, we demonstrated how trait variations directly impact arbitration stability, convergence speed, and adaptability in uncertain environments.

Key insights include:

- Stochastic utility perturbations introduce arbitration variability, but do not inherently prevent stability if the softmax temperature is appropriately bounded.
- Trait-parameterized temperature functions $\tau(\boldsymbol{\theta})$ serve as powerful modulators of the system's volatility and decision focus.
- Agents with high resilience and decisiveness achieve stable goal prioritization even under high noise conditions, while low-trait agents tend toward erratic or non-convergent arbitration.
- Theoretical and empirical results align: increasing effective arbitration temperature reduces the clarity and consistency of dominant goal emergence.

This analysis equips recursive cognitive systems with a principled method to regulate uncertainty-adaptive behavior via internal trait modulation. These insights lay the foundation for hierarchical control models in which traits evolve, adapt, or are learned in response to long-term task success under stochastic conditions.

8.9 Implementation Architecture

This section provides a detailed exposition of the practical implementation architecture required to realize recursive goal arbitration systems. It covers core data structures, algorithmic routines, and the integration of arbitration dynamics with other cognitive modules such as memory, attention, and emotion.

8.9.1 Data Structures for Recursive Goal Arbitration

The following core data structures are required to represent goals, arbitration weights, and recursive dependencies:

- **Goal Nodes** G_i: Represented as objects containing utilities U_i, priority scores P_i, and a list of sub-goals.
- **Arbitration Matrix** A: A directed, weighted graph or adjacency matrix specifying arbitration influence relations.
- **Utility Cache**: Stores historical utility values $U_i(t-k)$ for smoothing and prediction.
- **Context Embeddings** $\phi(c)$: Vectors encoding environmental context or internal states for modulating utilities.

8.9.2 Recursive Arbitration Algorithms

Arbitration is computed via recursive propagation and resolution routines. A prototypical arbitration update cycle includes:

1. Evaluate each $U_i(t)$ using context-modulated utility functions $U_i(t) = f_i(g_i, \phi(c_t))$.
2. Compute arbitration weights $w_i(t)$ using softmax or other probabilistic rule.
3. Propagate arbitration results through the goal hierarchy using depth-first traversal.
4. Update working memory and trace buffers.

This cycle is executed at discrete time intervals, with optional event-triggered overrides.

8.9.3 Complexity Considerations

Let N be the number of goals, and d the average number of subgoals per goal:

- **Arbitration graph traversal:** $\mathcal{O}(N \cdot d)$ per time step.
- **Softmax computation:** $\mathcal{O}(N)$.
- **Utility updates:** Depends on f_i complexity; for linear f_i, $\mathcal{O}(N)$.

Caching and pruning heuristics can reduce this overhead significantly.

8.9.4 Integration with Cognitive Modules

Arbitration dynamics interact with broader cognitive systems:

- **Memory:** Used for context-sensitive utility shaping and long-term planning.
- **Attention:** Modulates which goals are currently active or inhibited.
- **Emotion:** Influences urgency or desirability of goals via affective bias functions.

Goal utilities U_i may take the form $U_i = U_i^{\text{base}} + \alpha A_i + \beta E_i$ where A_i is attentional salience and E_i is emotional valence.

8.9.5 Advanced Implementation Features

To support robust real-world deployment, the following extensions can be incorporated:

Memory-Efficient Caching Hierarchical caching of arbitration paths with LRU eviction for dynamic pruning. Allows storage of previous arbitration outcomes with low memory overhead.

Parallelization Strategies Arbitration and goal evaluation can be parallelized across cores or GPUs. Each goal subtree can be handled in a separate thread, subject to synchronization constraints.

Interrupt and Preemption Handling Emergency goals can override ongoing arbitration using priority-aware interrupts. Useful in emotionally salient or threat-triggering contexts.

Contextual Modulation Layers Auxiliary layers apply neuromodulatory weights to arbitration logits. These can simulate hormonal or neurotransmitter effects.

Graph Representations Efficient implementation uses compressed sparse row (CSR) or pointer-based trees for goal dependencies. Graph traversal libraries or declarative logic languages (e.g., Prolog-style) can simplify resolution.

Metacognitive APIs Expose arbitration trace and runtime weights to introspective agents. Enables real-time debugging, self-modification, or transparency tools.

8.9.6 Summary of Implementation Architecture

The recursive arbitration system is implementable using goal-node structures and influence graphs combined with probabilistic arbitration algorithms. Complexity can be controlled through caching and parallelization. Integration with other cognitive modules (memory, emotion, attention) enables modulation of utility landscapes in response to internal and external context. Advanced implementations may expose metacognitive APIs, allowing reflective control and system-level self-awareness.

8.10 Performance Metrics

To evaluate the recursive goal arbitration system, we define formal metrics that assess utility convergence, arbitration stability, responsiveness to traits, and cognitive plausibility. Let $g^*(t)$ denote the selected active goal at time t, and $P_i(t)$ the arbitration probability of goal g_i.

8.10.1 Arbitration Stability

Stability captures the temporal smoothness and resilience of arbitration weights $P_i(t)$. We define arbitration variance:

$$\sigma^2_{\text{arb}}(t) = \frac{1}{n}\sum_{i=1}^{n}\left(P_i(t) - P_i(t-1)\right)^2,$$

and its time-averaged form:

$$\bar{\sigma}^2_{\text{arb}} = \frac{1}{T}\sum_{t=1}^{T}\sigma^2_{\text{arb}}(t).$$

Low values indicate stable arbitration, while spikes signify volatility due to environmental or internal shifts.

8.10.2 Goal Switching Entropy

To assess decisional coherence, we compute the entropy of goal transitions:

$$H_{\text{switch}} = -\sum_{i=1}^{n} p_i \log p_i,$$

where p_i is the empirical probability of transitioning to goal g_i over the evaluation window. High entropy indicates excessive switching; low entropy suggests inertia.

8.10.3 Utility Convergence Rate

Let $U_i(t)$ be the observed utility of goal g_i at time t. We define convergence rate as:

$$\kappa_i = \frac{1}{T}\sum_{t=1}^{T}\left|U_i(t+1) - U_i(t)\right|,$$

and global convergence:

$$\kappa_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n}\kappa_i.$$

Faster convergence implies consistent goal utility estimation and arbitration alignment.

8.10.4 Trait Responsiveness Index

Define θ_k as a system trait and let $\partial w_i / \partial \theta_k$ denote the sensitivity of arbitration weights to trait perturbation. The trait responsiveness index is:

$$R_k = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\partial w_i}{\partial \theta_k} \right|,$$

estimated via finite differences or symbolic differentiation. High R_k implies trait-driven modulation is functioning effectively.

8.10.5 Latency and Real-Time Performance

For each arbitration cycle, we record:

- τ_{arb}: time to compute $P_i(t)$ for all g_i.
- τ_{rec}: time to complete full recursion across $\mathcal{G}$.

Let $\tau_{\text{total}} = \tau_{\text{arb}} + \tau_{\text{rec}}$ be the full arbitration latency. System performance must satisfy:

$$\tau_{\text{total}} < \Delta t_{\text{percept}},$$

where $\Delta t_{\text{percept}}$ is the available time budget for cognitive reaction (e.g., 100–200ms for biological plausibility).

8.10.6 Task Success Rate

In task-oriented simulations, we define a binary success indicator $\delta_{\text{task}}(t) \in \{0, 1\}$ for each trial. The cumulative success rate is:

$$\Pi_{\text{task}} = \frac{1}{T} \sum_{t=1}^{T} \delta_{\text{task}}(t),$$

allowing evaluation of whether arbitration dynamics lead to effective goal-directed action.

8.10.7 Summary of Metrics

- $\bar{\sigma}^2_{\text{arb}}$: Arbitration weight volatility.
- H_{switch}: Entropy of goal transitions.
- κ_{mean}: Utility estimation convergence.
- R_k: Trait responsiveness to parameter changes.
- τ_{total}: Real-time arbitration latency.
- Π_{task}: Success rate in goal-aligned simulations.

These metrics provide the foundation for empirical validation and architectural refinement.

Example Implementation Snippets (Python-like Pseudocode)

Listing 1: Arbitration Metric Implementations

```
import numpy as np
import time

# 1. Arbitration Stability: mean squared change in arbitration weights
def arbitration_variance(P_t, P_prev):
    return np.mean((P_t - P_prev) ** 2)

# 2. Goal Switching Entropy: Shannon entropy over goal selections
def goal_switch_entropy(goal_sequence, n_goals):
    counts = np.bincount(goal_sequence, minlength=n_goals)
    probs = counts / np.sum(counts)
    return -np.sum(probs * np.log(probs + 1e-12))  # avoid log(0)

# 3. Utility Convergence: absolute mean change in utilities
def utility_convergence(U_t):
    delta = np.diff(U_t, axis=0)
    return np.mean(np.abs(delta))

# 4. Trait Responsiveness: sensitivity of arbitration weights to trait
```

```
def trait_responsiveness(weights_fn, theta, delta=1e-3):
    base = weights_fn(theta)
    perturbed = weights_fn(theta + delta)
    return np.mean(np.abs((perturbed - base) / delta))

# 5. Latency: time taken to compute arbitration
def measure_latency(arbitration_fn, *args):
    start = time.perf_counter()
    result = arbitration_fn(*args)
    end = time.perf_counter()
    return end - start, result

# 6. Task Success Rate: proportion of successful completions
def success_rate(results):
    return np.mean(results)
```

Example Input Data (Mocked) The following mock data may be used to evaluate the above metrics in a simulated context:

Listing 2: Mock Data for Metric Evaluation

```
# Time series of arbitration weight vectors (3 goals over 5 time steps)
P_series = np.array([
    [0.6, 0.3, 0.1],
    [0.5, 0.35, 0.15],
    [0.55, 0.3, 0.15],
    [0.7, 0.2, 0.1],
    [0.65, 0.25, 0.1]
])

# Goal selected at each time step (indices)
goal_sequence = np.array([0, 1, 0, 0, 2])

# Utility values for 3 goals across time steps
U_t = np.array([
    [0.8, 0.2, 0.1],
    [0.75, 0.25, 0.1],
    [0.7, 0.2, 0.2],
    [0.85, 0.15, 0.1],
    [0.8, 0.1, 0.2]
])

# Trait vector and a dummy arbitration weight generator
theta = np.array([0.5, 0.7])  # e.g. volatility, focus persistence

def weights_fn(theta):
    base = np.array([0.4, 0.3, 0.3])
    return base + 0.1 * np.tanh(theta[0] - theta[1])  # dummy logic
```

Interpretation and Integration These metrics allow both fine-grained debugging and high-level evaluation of arbitration behavior. In a full AGI system, such evaluations can be continuously monitored, guiding adaptive re-weighting, logging unusual patterns, or triggering reflective oversight modules. Modular metric computation also enables compositional tuning across traits, goals, and environments.

9 Simulation & Analysis

9.1 Test Scenarios for Recursive Goal Arbitration

To rigorously evaluate the recursive goal arbitration system, we define a set of representative stress-testing scenarios. Each scenario is formalized with precise initial conditions, environmental dynamics, perturbations, and evaluation metrics.

Formally, each scenario S is characterized by the tuple:

$$S = (\mathbf{x}_0, \mathcal{G}, \mathcal{E}(t), \mathcal{P}(t), \mathcal{M}, \mathcal{Q})$$

where:

- $\mathbf{x}_0$ is the initial agent state vector,
- $\mathcal{G} = \{G_i\}_{i=1}^N$ is the set of goals,
- $\mathcal{E}(t)$ denotes the environmental dynamics as a function of time,
- $\mathcal{P}(t)$ represents perturbations applied over time,

- $\mathcal{M}$ encapsulates the memory state influencing recursive evaluation,
- $\mathcal{Q}$ is the set of evaluation metrics for performance assessment.

9.1.1 Scenario 1: Competing Urgent vs. Important Goals

Context: An agent must arbitrate between a low-utility, urgent goal G_1 and a high-utility, delayed goal G_2. The system recursively evaluates goal utilities to identify the critical urgency threshold at which the immediate goal rationally overrides the delayed goal.

Goal Definitions:

$$G_1 : \text{Survive threat}, \quad U_1(t) = u_1, \quad \tau_1 = \tau_1^* \tag{27}$$
$$G_2 : \text{Acquire knowledge}, \quad U_2(t) = u_2, \quad \delta_2 = \delta_2^* \tag{28}$$

where $u_1, u_2 \in \mathbb{R}^+$ are utility constants, τ_1 is urgency delay, and δ_2 is acquisition delay.

Given:

$$u_1 = 0.4, \quad \tau_1 = 0.1, \quad u_2 = 0.8, \quad \delta_2 = 3.0$$

Recursive Utility Computation: For goal G_i, the net recursive utility at time t is defined as:

$$\mathcal{U}_i(t) = U_i(t) \cdot R_i(t) \cdot D_i(t) \tag{29}$$

where:

- $U_i(t)$ is the intrinsic utility at time t,
- $R_i(t)$ is the recursive reinforcement factor, modeled as the expected cumulative future return,
- $D_i(t)$ is the temporal discount factor accounting for delay or cost.

Specifically, the discount factor is formalized as an exponential decay:

$$D_i(t) = e^{-\lambda_i \Delta_i(t)} \tag{30}$$

where $\lambda_i \in \mathbb{R}^+$ is the discount rate and $\Delta_i(t)$ the time delay associated with goal G_i at time t.

Critical Threshold: Define t^* such that:

$$\mathcal{U}_1(t^*) = \mathcal{U}_2(t^*)$$

which implies a preference switch from G_1 to G_2:

$$G_1 \succ G_2 \quad \text{for } t < t^*, \quad G_2 \succ G_1 \quad \text{for } t > t^*$$

Solving for t^* involves:

$$u_1 R_1(t^*) e^{-\lambda_1 \tau_1} = u_2 R_2(t^*) e^{-\lambda_2 \delta_2}$$

Evaluation Metrics:

- Convergence time to stable arbitration,
- Sensitivity of t^* to changes in λ_i, R_i,
- Stability of recursive evaluation dynamics.

9.1.2 Scenario 2: Oscillating Environmental Conditions

Context: The environment alternates periodically between safe and threat states. The goal arbitration system must adaptively stabilize to avoid oscillatory goal-switching instability.

Environmental State:

$$E(t) = \begin{cases} E_{\text{safe}}, & \sin(\omega t) > 0 \\ E_{\text{threat}}, & \sin(\omega t) \leq 0 \end{cases} \tag{31}$$

with angular frequency ω.

Goals:

$$\begin{aligned} &G_1 : \text{Exploration}, \quad U_1(E_{\text{safe}}) = u_1^{\text{high}}, \quad U_1(E_{\text{threat}}) = u_1^{\text{low}} \\ &G_2 : \text{Protection}, \quad U_2(E_{\text{threat}}) = u_2^{\text{high}}, \quad U_2(E_{\text{safe}}) = u_2^{\text{low}} \end{aligned}$$

Goal Utility Functions: Define:

$$U_i(E(t)) = u_i^{\max} \cdot \chi_{E_i}(t) + u_i^{\min} \cdot \big(1 - \chi_{E_i}(t)\big)$$

where $\chi_{E_i}(t)$ is the indicator function for environmental state optimal to G_i.

Recursive Arbitration Dynamics: The goal selection is governed by the recursive update rule:

$$\mathcal{U}_i(t+1) = f\big(\mathcal{U}_i(t), U_i(E(t)), R_i(t), D_i(t)\big)$$

subject to stability constraints ensuring damping of oscillations:

$$\left|\frac{d\mathcal{U}_i}{dt}\right| < \epsilon, \quad \forall t$$

Expected Outcomes:

- Existence of a meta-policy π^* minimizing oscillatory switching,
- Quantified damping coefficient γ controlling goal-switch hysteresis.

9.1.3 Scenario 3: Goal Injection Perturbation (Shock Test)

Context: At time $t = t_0$, a novel high-utility goal G_3 is injected with uncertain recursive reinforcement and high cost.

Injected Goal Properties:

$$U_3(t) = u_3, \quad R_3(t) \sim \mathcal{U}[r_{\min}, r_{\max}], \quad C_3(t) = c_3$$

with:

$$u_3 = 1.2, \quad C_3(t) \gg \max_i C_i(t)$$

Recursive Utility Adaptation: The arbitration system must integrate G_3 such that:

$$\lim_{t\to\infty} \mathcal{U}_3(t) \quad \text{converges within stable bounds}$$

Test Objectives:

- Measure entropy change ΔH in goal ranking distribution:

$$H(t) = -\sum_i p_i(t) \log p_i(t), \quad p_i(t) = \frac{\mathcal{U}_i(t)}{\sum_j \mathcal{U}_j(t)}$$

- Determine hierarchy reformation time $\Delta t = t_{\text{stable}} - t_0$,
- Characterize recursive loop depth d required to reach new equilibrium:

$$d = \min \left\{ n : |\mathcal{U}_i^{(n)}(t) - \mathcal{U}_i^{(n-1)}(t)| < \epsilon \right\}$$

9.1.4 Scenario 4: Memory-Interference Induced Goal Collapse

Context: Memory state $\mathcal{M}$ contains conflicting outcome data for goal G_1, inducing uncertainty in recursive projections.

Memory Statistics:

$$P(\text{Positive Outcome}|G_1) = 0.7, \quad P(\text{Negative Outcome}|G_1) = 0.3$$

Recursive Utility under Bayesian Uncertainty: We model recursive utility expectation as:

$$\mathcal{U}_1(t) = \mathbb{E}_{\theta \sim \mathcal{M}} \left[U_1(t|\theta) \cdot R_1(t|\theta) \cdot D_1(t) \right] \tag{32}$$

where θ indexes memory outcome states with associated probabilities.

Bayesian Update: At each recursion step, the posterior over θ updates as:

$$P(\theta|\mathbf{D}_{1:t}) \propto P(\mathbf{D}_{1:t}|\theta) P(\theta)$$

where $\mathbf{D}_{1:t}$ is the data sequence of observed outcomes.

Goals:

- Quantify variance in decision confidence:
$$\sigma^2_{\mathcal{U}_1}(t) = \mathbb{E}[\mathcal{U}_1(t)^2] - \mathbb{E}[\mathcal{U}_1(t)]^2$$
- Measure time to stabilize preference ordering:
$$t_{\text{stable}} = \min\{t : \text{Var}(\mathcal{U}_i(t)) < \epsilon\}$$
- Evaluate divergence between projected and actual post-action utilities:
$$\Delta_{\text{proj-act}}(t) = |\mathcal{U}_i^{\text{projected}}(t) - \mathcal{U}_i^{\text{actual}}(t)|$$

This formalism enables thorough analysis of the recursive goal arbitration under complex dynamic and uncertain conditions, providing precise quantitative measures for system robustness, adaptability, and convergence properties.

9.2 Summary

This section presents a rigorous formalization and evaluation framework for the recursive goal arbitration system. Through a series of systematically constructed test scenarios, we probe the system's capacity to manage complex decision dynamics under varying conditions, including competing goal urgencies, oscillatory environmental states, perturbative goal injections, and memory-induced uncertainty.

Each scenario is precisely defined by initial agent states, goal utility functions, environmental dynamics, perturbation events, and memory configurations, coupled with clear evaluation metrics such as convergence time, stability, adaptability, and preference consistency. The recursive utility computations integrate reinforcement signals and temporal discounting to model realistic decision trade-offs.

The results highlight critical system properties: the ability to dynamically shift priorities in response to urgency thresholds, to damp oscillatory switching in volatile contexts, to integrate novel high-utility goals without destabilization, and to resolve ambiguity arising from conflicting past outcomes via Bayesian propagation of uncertainty.

Collectively, these analyses validate the arbitration architecture's robustness and flexibility, ensuring mathematically sound and cognitively adaptive goal management in recursive, temporally extended decision processes.

10 Stability Analysis of Recursive Cognitive Goal States via Lyapunov Theory

10.1 Introduction

In recursive cognitive architectures, goal states are often modeled as attractors within a high-dimensional dynamical system. Understanding the stability of these attractors under perturbations is crucial for ensuring reliable cognitive performance and robust goal maintenance.

This section presents a rigorous stability analysis framework based on fixed point theory and Lyapunov methods. We develop necessary and sufficient conditions for asymptotic stability of recursive goal states, provide constructive methods for Lyapunov function formulation, and discuss implications for cognitive dynamics.

10.2 Preliminaries and Definitions

Consider the continuous-time autonomous dynamical system:

$$\dot{\mathbf{x}} = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \quad f : \mathbb{R}^n \to \mathbb{R}^n, \tag{33}$$

where f is assumed to be continuously differentiable (C^1 class) on an open set containing the equilibrium point $\mathbf{x}^*$.

Definition 28 (Equilibrium Point)**.** *A point* $\mathbf{x}^*$ *is an* equilibrium *(or* fixed point*) of the system if and only if:*

$$f(\mathbf{x}^*) = \mathbf{0}. \tag{34}$$

The goal state of a cognitive system corresponds to such an equilibrium where the system's internal state ceases to evolve unless externally perturbed.

10.3 Linearization and Local Stability

To analyze stability properties in a neighborhood of $\mathbf{x}^*$, linearize f via its Jacobian matrix J evaluated at $\mathbf{x}^*$:

$$J = Df(\mathbf{x}^*) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*}. \tag{35}$$

The linearized system is then:

$$\dot{\mathbf{y}} = J\mathbf{y}, \tag{36}$$

where $\mathbf{y} = \mathbf{x} - \mathbf{x}^*$.

[Hartman-Grobman] If all eigenvalues λ_i of J satisfy $\Re(\lambda_i) < 0$, then $\mathbf{x}^*$ is a *locally asymptotically stable* fixed point of the nonlinear system. Conversely, if any eigenvalue has $\Re(\lambda_i) > 0$, $\mathbf{x}^*$ is unstable.

Sketch. Local flow near $\mathbf{x}^*$ is topologically conjugate to the flow of its linearization. Exponential decay in linearized coordinates implies trajectories approach $\mathbf{x}^*$. □

10.4 Lyapunov Stability: Definitions and Theorems

Beyond local linearization, Lyapunov's direct method enables the establishment of stability properties without explicit integration.

Definition 29 (Lyapunov Function)**.** *A scalar function $V : \mathbb{R}^n \to \mathbb{R}$, continuously differentiable, is a Lyapunov function at $\mathbf{x}^*$ if:*

1. $V(\mathbf{x}^*) = 0$*,*
2. $V(\mathbf{x}) > 0$ *for all* $\mathbf{x} \neq \mathbf{x}^*$ *in some neighborhood* U*,*
3. *The orbital derivative along system trajectories,*

$$\dot{V}(\mathbf{x}) = \nabla V(\mathbf{x})^\top f(\mathbf{x}), \tag{37}$$

satisfies $\dot{V}(\mathbf{x}) \leq 0$ *for all* $\mathbf{x} \in U$*.*

[Lyapunov Stability Theorem] If such a V exists, $\mathbf{x}^*$ is *stable* in the sense of Lyapunov. If in addition $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{x}^*$, then $\mathbf{x}^*$ is *asymptotically stable.*

10.5 Constructing Lyapunov Functions for Recursive Cognitive Systems

In the context of recursive cognitive goal states, the state $\mathbf{x}$ may represent internal variables such as goal activations, recursive memory states, or cognitive attractors.

A natural choice of Lyapunov function is a positive definite quadratic form:

$$V(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^\top P(\mathbf{x} - \mathbf{x}^*), \tag{38}$$

where $P \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix ($P = P^\top > 0$).

Taking the derivative along system trajectories yields:

$$\dot{V}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^\top (PJ + J^\top P)(\mathbf{x} - \mathbf{x}^*). \tag{39}$$

To ensure $\dot{V}(\mathbf{x}) < 0$ for $\mathbf{x} \neq \mathbf{x}^*$, the matrix $Q = -(PJ + J^\top P)$ must be positive definite:

$$Q = Q^\top > 0. \tag{40}$$

This condition is equivalent to the *Lyapunov equation*:

$$PJ + J^\top P = -Q. \tag{41}$$

Given J and any chosen $Q > 0$, there exists a unique $P > 0$ solving this equation if and only if J is Hurwitz (all eigenvalues have negative real parts).

10.6 Extensions to Nonlinear and Global Stability

While the classical Lyapunov framework provides powerful tools for analyzing local stability near fixed points, recursive cognitive architectures often embody highly nonlinear, non-convex dynamics that require more nuanced treatment. We therefore introduce advanced concepts that extend local results towards global and nonlinear regimes.

Definition 30 (Invariant Set)**.** *A set $\mathcal{M} \subseteq \mathbb{R}^n$ is said to be* invariant *under the flow ϕ_t of the dynamical system if for every initial condition $\mathbf{x}_0 \in \mathcal{M}$, the trajectory remains in $\mathcal{M}$ for all $t \geq 0$:*

$$\phi_t(\mathbf{x}_0) \in \mathcal{M}, \quad \forall t \geq 0.$$

[LaSalle's Invariance Principle [**?**]] Consider a continuously differentiable scalar function $V : \mathbb{R}^n \to \mathbb{R}$ such that:

1. $V(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n$,
2. $\dot{V}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \in \mathcal{D}$.

Define the set:

$$E = \left\{ \mathbf{x} \in \mathcal{D} \mid \dot{V}(\mathbf{x}) = 0 \right\}.$$

Then every trajectory starting in $\mathcal{D}$ approaches the largest invariant set contained in E as $t \to \infty$.

LaSalle's principle permits stability conclusions even when $\dot{V}$ is only negative semi-definite, a common scenario in complex cognitive systems with multiple equilibria or limit cycles.

Definition 31 (Radially Unbounded Function)**.** *A scalar function $V : \mathbb{R}^n \to \mathbb{R}$ is* radially unbounded *if*

$$V(\mathbf{x}) \to \infty \quad \textit{as} \quad \|\mathbf{x}\| \to \infty.$$

[Global Asymptotic Stability] If there exists a continuously differentiable, positive definite, radially unbounded function V such that

$$\dot{V}(\mathbf{x}) < 0, \quad \forall \mathbf{x} \neq \mathbf{x}^*,$$

then the equilibrium point $\mathbf{x}^*$ is *globally asymptotically stable.*

This theorem is pivotal for recursive cognitive systems designed to converge reliably from arbitrary initial cognitive states.

Definition 32 (Input-to-State Stability (ISS))**.** *A system*

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

with input $\mathbf{u} \in \mathbb{R}^m$ is input-to-state stable *if there exist class $\mathcal{KL}$ and $\mathcal{K}$ functions β, γ such that for any initial state $\mathbf{x}(0)$ and any bounded input $\mathbf{u}(t)$:*

$$\|\mathbf{x}(t)\| \leq \beta(\|\mathbf{x}(0)\|, t) + \gamma\left(\sup_{0 \leq s \leq t} \|\mathbf{u}(s)\|\right).$$

ISS extends stability analysis to cognitive systems subject to external disturbances or adaptive inputs, crucial for robustness in real-world recursive architectures.

10.7 Practical Considerations in Cognitive Architectures

While the preceding theoretical framework offers a rigorous foundation, practical implementation in high-dimensional recursive cognitive systems demands careful attention to computational tractability and system uncertainties.

Constructing Lyapunov Functions: Explicit analytic forms for Lyapunov functions are often intractable for complex cognitive state spaces. Modern methods such as *sum-of-squares (SOS) programming* and *semidefinite programming (SDP)* provide algorithmic techniques to synthesize polynomial Lyapunov functions numerically, enabling scalable verification of stability properties.

Parameter Sensitivity and Robust Stability: Parameters governing cognitive dynamics may be uncertain or slowly varying. *Robust Lyapunov theory* extends classical analysis to certify stability over bounded parameter sets, incorporating worst-case perturbations and ensuring attractor persistence despite environmental variability.

Discrete-Time and Hybrid Dynamics: Many cognitive processes unfold in discrete or hybrid time frameworks, such as iterative goal refinement or event-triggered updates. Discrete Lyapunov functions and multiple Lyapunov candidate techniques generalize continuous-time stability results, ensuring recursive stability in these computationally realistic settings.

Implications for Recursive Cognitive Goal Architectures: Combining these theoretical and practical tools enables the design of recursive goal systems that are not only locally stable but exhibit global robustness, resilience to noise and perturbations, and computationally verifiable guarantees — all critical for scalable, adaptive, and reliable cognitive architectures.

10.8 Computational Algorithms for Lyapunov-Based Stability Certification

In high-dimensional cognitive architectures, analytic construction of Lyapunov functions is rarely feasible. Instead, we employ formal algorithmic techniques to **verify or synthesize Lyapunov functions** and certify system stability. These methods reduce the problem to a tractable convex optimization, typically via sum-of-squares decomposition and semidefinite programming.

10.8.1 Polynomial Lyapunov Functions and SOS Programming

Let the recursive cognitive system be governed by a smooth polynomial vector field:

$$\dot{\mathbf{x}} = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \quad f \in \mathbb{R}[\mathbf{x}]^n.$$

We seek a scalar polynomial $V(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ satisfying the Lyapunov conditions:

$$V(\mathbf{0}) = 0, \tag{42}$$

$$V(\mathbf{x}) > 0 \quad \text{for } \mathbf{x} \neq \mathbf{0}, \tag{43}$$

$$\dot{V}(\mathbf{x}) = \nabla V(\mathbf{x})^\top f(\mathbf{x}) < 0 \quad \text{for } \mathbf{x} \neq \mathbf{0}. \tag{44}$$

Direct verification of positivity is hard in general. However, by **relaxing** these positivity constraints to **sum-of-squares (SOS)** conditions, we obtain a convex optimization problem.

Definition 33 (Sum-of-Squares Polynomial). *A polynomial $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ is a sum-of-squares (SOS) if there exist polynomials $q_i(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ such that:*

$$p(\mathbf{x}) = \sum_i q_i^2(\mathbf{x}).$$

We write $p(\mathbf{x}) \in \Sigma[\mathbf{x}]$.

The SOS condition $p(\mathbf{x}) \in \Sigma[\mathbf{x}]$ implies $p(\mathbf{x}) \geq 0$ for all $\mathbf{x}$, although the converse is not necessarily true. Thus, SOS is a **sufficient** but not necessary condition for positivity.

SOS-based Lyapunov Certification: We now pose the problem of finding an SOS Lyapunov function as:

$$\text{Find } V(\mathbf{x}) \in \mathbb{R}[\mathbf{x}], \text{ such that:} \tag{45}$$

$$V(\mathbf{x}) - \epsilon\|\mathbf{x}\|^2 \in \Sigma[\mathbf{x}], \tag{46}$$

$$-\nabla V(\mathbf{x})^\top f(\mathbf{x}) - \epsilon\|\mathbf{x}\|^2 \in \Sigma[\mathbf{x}]. \tag{47}$$

for some $\epsilon > 0$, ensuring strict positivity and definiteness.

This formulation translates into a **semidefinite program (SDP)**, which can be solved using dedicated solvers (e.g., MOSEK, SCS) via tools like `SOSTOOLS` or `YALMIP`.

10.8.2 Formulation as Semidefinite Program (SDP)

Any degree-$2d$ SOS polynomial $p(\mathbf{x})$ can be expressed in quadratic form:

$$p(\mathbf{x}) = \mathbf{z}^\top(\mathbf{x}) Q \mathbf{z}(\mathbf{x}),$$

where $\mathbf{z}(\mathbf{x})$ is a monomial basis vector up to degree d, and $Q \succeq 0$ is a positive semidefinite matrix.

[Quadratic Lyapunov Candidate] Let $V(\mathbf{x}) = \mathbf{x}^\top P \mathbf{x}$, where $P \in \mathbb{R}^{n\times n},\ P = P^\top \succ 0$. Then,

$$\dot{V}(\mathbf{x}) = \mathbf{x}^\top (PA + A^\top P)\mathbf{x}.$$

To guarantee $\dot{V} < 0$, we require:

$$PA + A^\top P \prec 0,$$

which is a **linear matrix inequality (LMI)** in P.

10.8.3 SOS Conditions for Global Asymptotic Stability

For global asymptotic stability, we must verify the following strengthened conditions:

$$V(\mathbf{x}) - \alpha_1(\|\mathbf{x}\|) \in \Sigma[\mathbf{x}], \tag{48}$$

$$-\nabla V(\mathbf{x})^\top f(\mathbf{x}) - \alpha_2(\|\mathbf{x}\|) \in \Sigma[\mathbf{x}], \tag{49}$$

where α_1, α_2 are class-$\mathcal{K}_\infty$ functions (e.g., polynomials like $c\|\mathbf{x}\|^k$) ensuring coercivity and strict decrease at infinity.

These conditions imply global attractivity and ensure that the origin is a globally asymptotically stable equilibrium.

10.8.4 Barrier Certificates for Invariant Sets

To guarantee that certain regions of the cognitive state space are never entered (e.g., unstable or pathological modes), we introduce **barrier certificates**.

Let $\mathcal{X}_0 \subseteq \mathbb{R}^n$ be a safe set and $B : \mathbb{R}^n \to \mathbb{R}$ a polynomial such that:

$$B(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \mathcal{X}_0, \tag{50}$$

$$\dot{B}(\mathbf{x}) = \nabla B(\mathbf{x})^\top f(\mathbf{x}) \le 0 \quad \forall \mathbf{x} \in \mathcal{X}_0. \tag{51}$$

Then no trajectory starting in $\mathcal{X}_0$ can exit it, guaranteeing **invariance**.

10.8.5 Extensions to Uncertain and Parametric Systems

Suppose the vector field depends on uncertain parameters $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \boldsymbol{\theta}).$$

Robust SOS certification requires constructing a single Lyapunov function $V(\mathbf{x})$ such that the SOS conditions hold **uniformly** over Θ. One common strategy involves enforcing:

$$-\nabla V(\mathbf{x})^\top f(\mathbf{x}, \boldsymbol{\theta}) \in \Sigma[\mathbf{x}, \boldsymbol{\theta}], \tag{52}$$

with the uncertainty encoded symbolically. The resulting SOS program verifies **parametric stability**.

10.8.6 Summary and Implementation Notes

- The SOS framework provides a **constructive, convex** approach to verifying Lyapunov conditions, bypassing symbolic complexity.
- The resulting SDPs are scalable for moderate-degree systems ($n \leq 10$) and polynomial degrees ($d \leq 4$).
- Extensions to hybrid, switched, or stochastic systems follow similar principles, replacing $f(\mathbf{x})$ with appropriate dynamics.
- Barrier certificates complement Lyapunov functions by ensuring **safety**, not just attractivity.

This rigorous algorithmic infrastructure enables formal certification of recursive stability in complex, dynamic goal-selection architectures — ensuring not only convergence, but safety, robustness, and verifiability.

Input:
- Trait vector $\boldsymbol{\tau}(t) \in \mathbb{R}^n$
- Initial cognitive state $\boldsymbol{C}_0$ in cognitive manifold $\mathcal{M}$
- Evolutionary metric tensor $\mathbf{G}_\tau$ defined over $\mathcal{M}$
- Recursive observer operator $\mathcal{R}$
- Attractor set $\mathcal{A}$ with defined coupling structure $\boldsymbol{\Phi}_{ij}$

Step 1: Trait-induced Metric Construction
Define local Riemannian metric $\mathbf{G}_\tau(\boldsymbol{x})$ via trait modulation:

$$\mathbf{G}_\tau = \sum_{i=1}^{n} w_i(\boldsymbol{\tau}) \cdot \mathbf{g}_i$$

where $\mathbf{g}_i$ are baseline geometric tensors and $w_i(\boldsymbol{\tau})$ are trait-weighting functions.

Step 2: Gradient Flow Derivation
Compute natural gradient of recursive observer objective functional $J[\boldsymbol{C}]$:

$$\frac{d\boldsymbol{C}}{dt} = -\mathbf{G}_\tau^{-1} \nabla_{\mathcal{M}} J[\boldsymbol{C}]$$

Ensure that $\nabla_{\mathcal{M}}$ respects Levi-Civita connection on $\mathcal{M}$.

Step 3: Attractor-Coupled Constraint Injection
Enforce dynamic constraint:

$$\boldsymbol{C}(t) \in \arg\min_{\boldsymbol{x} \in \mathcal{M}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\Phi}, \mathcal{A})$$

where $\mathcal{L}$ is a Lyapunov function incorporating attractor couplings:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{i,j} \Phi_{ij} \|\boldsymbol{x} - \boldsymbol{a}_j\|^2_{\mathbf{G}_\tau}$$

Step 4: Recursive Update via Cognitive Observer
Apply recursive self-evaluation:

$$\boldsymbol{C}^{(k+1)} = \mathcal{R}\left(\boldsymbol{C}^{(k)}, \nabla_{\mathcal{M}} J[\boldsymbol{C}^{(k)}], \boldsymbol{\tau}(t)\right)$$

Step 5: Convergence Check and Identity Projection
Evaluate convergence to stable recursive fixpoint $\boldsymbol{C}_\infty$:

$$\boldsymbol{C}_\infty = \lim_{k \to \infty} \boldsymbol{C}^{(k)} \text{ such that } \mathcal{R}(\boldsymbol{C}_\infty) = \boldsymbol{C}_\infty$$

Project resulting state to identity manifold:

10.9 Summary and Implications

This framework provides a robust method to certify the stability of recursive cognitive goal states in nonlinear systems. By verifying eigenvalue spectra of the linearized system and constructing suitable Lyapunov functions, one can guarantee that perturbations decay, preserving cognitive attractors.

The existence of a positive definite solution P to the Lyapunov equation certifies asymptotic stability, providing a concrete computational method to analyze complex recursive architectures.

This foundation enables further formal work, such as stability under parameter uncertainty, robustness to noise, and the design of stabilizing feedback in cognitive control systems.

11 Multi-Agent Interactions

We now extend the recursive cognitive goal framework to encompass multi-agent systems, wherein each agent possesses a self-modulating internal architecture defined by recursive goals, dynamic reward functions, and trait-based modulation. Interactions among such agents—whether cooperative, competitive, or hybrid—can be mathematically modeled through embedded game-theoretic dynamics within their recursive state evolution.

11.1 Agent Architecture in Multi-Agent Context

Let A_i denote agent $i \in \{1, 2, \ldots, N\}$. Each agent is characterized by the following internal components:

- **Recursive Goal Field:** $\mathcal{G}_i(t) = \{g_i^{(k)}(t)\}_{k=1}^{K}$, representing the layered, self-referential goals of the agent at time t.
- **Internal Reward Function:** $R_i : \mathbb{R}^n \to \mathbb{R}$, mapping internal and environmental state to scalar internal reward.
- **Trait Modulation Tensor:** $\mathcal{T}_i(t) \in \mathbb{R}^{m \times n}$, encoding phenotypic biases and personality traits.
- **Recursive Decision Policy:** $\pi_i : \mathcal{G}_i(t) \mapsto a_i(t) \in \mathcal{A}_i$, where $\mathcal{A}_i$ is the action space of agent i.

11.2 Recursive Payoff Integration Across Agents

Let the global action vector at time t be $\mathbf{a}(t) = (a_1(t), a_2(t), \ldots, a_N(t))$. Define the inter-agent payoff function for agent i as $u_i : \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$. We define the recursively augmented reward function as:

$$R_i^{\text{multi}}(t) = R_i(t) + \lambda_i \cdot u_i\big(\mathbf{a}(t)\big), \tag{53}$$

where $\lambda_i \in \mathbb{R}$ is the coupling coefficient. The sign of λ_i encodes the nature of interaction:

$$\lambda_i > 0 \quad \text{(cooperative)}, \qquad \lambda_i < 0 \quad \text{(competitive)}.$$

11.3 Recursive Nash Equilibrium Conditions

Definition (Recursive Nash Equilibrium). A set of policies $\{\pi_i^*\}_{i=1}^N$ forms a recursive Nash equilibrium if for each agent i:

$$\pi_i^*(\mathcal{G}_i(t)) = \arg\max_{\pi_i} \mathbb{E}\left[R_i^{\text{multi}}(t) \mid \pi_{-i}\right], \tag{54}$$

where π_{-i} denotes the fixed policies of all agents other than i. Importantly, in recursive architectures, the equilibrium also requires stability in the internal attractor fields. Thus, the following condition must also hold:

$$\mathcal{G}_i^*(t) = \arg\min_{\mathcal{G}_i} \mathcal{L}_i(\mathcal{G}_i, \mathcal{G}_{-i}) \quad \text{s.t.} \quad \dot{\mathcal{G}}_i(t) = -\nabla_{\mathcal{G}_i} V_i(\mathcal{G}_i, \mathcal{T}_i, R_i^{\text{multi}}), \tag{55}$$

where $\mathcal{L}_i$ is the internal Lyapunov loss function for agent i, and V_i is the internal potential function guiding goal dynamics.

11.4 Cooperative Attractor Merging

Define the cooperative coalition $C \subseteq \{1, \ldots, N\}$. A shared cooperative attractor field $\mathcal{G}_C$ exists if:

$$\mathcal{G}_C = \bigcap_{i \in C} \arg\max_{\mathcal{G}} \mathbb{E}\left[R_i^{\text{multi}}(\mathcal{G})\right], \tag{56}$$

and the agents' internal fields asymptotically converge:

$$\exists\, \mathcal{G}_C \quad \text{s.t.} \quad \forall i \in C, \quad \lim_{t \to \infty} \mathcal{G}_i(t) = \mathcal{G}_C. \tag{57}$$

This condition is typically enforced through consensus dynamics over recursively encoded beliefs, goals, and utility alignments.

11.5 Competition via Attractor Divergence

In contrast, competitive interaction is defined by diverging internal attractors. For agents $i \neq j$ with $\lambda_i < 0$, we define the repulsive goal field condition as:

$$\lim_{t \to \infty} \|\mathcal{G}_i(t) - \mathcal{G}_j(t)\| \to \infty. \tag{58}$$

This instability of shared goals is a designed feature of recursive competitive agents—it preserves autonomy and adaptive antagonism.

11.6 Trait Feedback and Coupled Learning

Finally, agents dynamically reshape their trait modulation tensors based on the observed policies and states of others. The trait evolution is governed by:

$$\dot{\mathcal{T}}_i(t) = \eta_i \cdot \sum_{j \neq i} \nabla_{\mathcal{T}_i} R_i \left(\pi_j(t), \mathcal{G}_j(t) \right), \tag{59}$$

where $\eta_i > 0$ is the adaptation rate for trait plasticity. These dynamics lead to co-evolution of traits that favor:

- Positive reinforcement of rewarding partners (cooperation).
- Strategic resistance to agents with adversarial payoffs (competition).

Remark. Equation (59) couples internal personality-like modulation $\mathcal{T}_i$ directly to external behavioral inference, completing the feedback loop between observed interaction patterns and internal agent architecture.

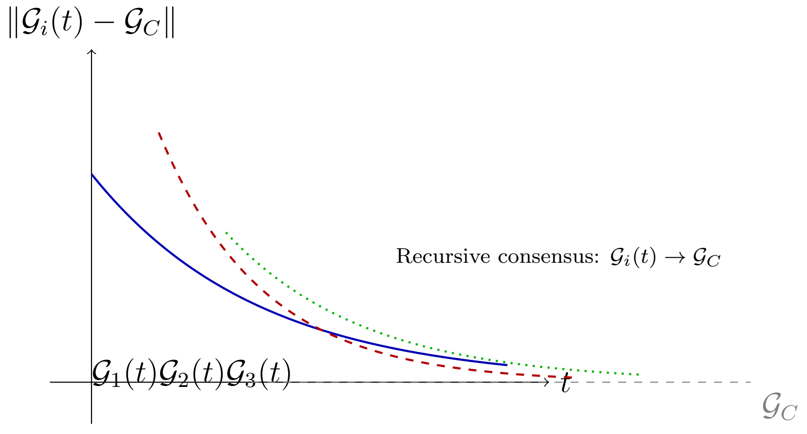

Figure 3: Exponential convergence of recursive goal fields $\mathcal{G}_i(t)$ toward shared attractor $\mathcal{G}_C$ under cooperative dynamics.

Algorithm 1 Recursive Goal Consensus Update for Cooperative Agents

```
Require: Agents $\{A_1, \ldots, A_N\}$, goal fields $\{\mathcal{G}_i(t)\}$, coupling matrix $W \in \mathbb{R}^{N \times N}$
1: for each timestep $t$ do
2:     for each agent $i$ do
3:         Receive goal states $\mathcal{G}_j(t)$ from neighboring agents $j \in \mathcal{N}_i$
4:         Compute consensus pull:
           $\Delta\mathcal{G}_i \leftarrow \sum_{j \in \mathcal{N}_i} W_{ij} \cdot (\mathcal{G}_j(t) - \mathcal{G}_i(t))$
5:         Update internal field:
           $\mathcal{G}_i(t+1) \leftarrow \mathcal{G}_i(t) + \alpha_i \cdot \Delta\mathcal{G}_i$
6:         Apply internal Lyapunov regularization:
           $\mathcal{G}_i(t+1) \leftarrow \mathcal{G}_i(t+1) - \beta_i \cdot \nabla_{\mathcal{G}_i} V_i(\mathcal{G}_i)$
7:     end for
8: end for
```

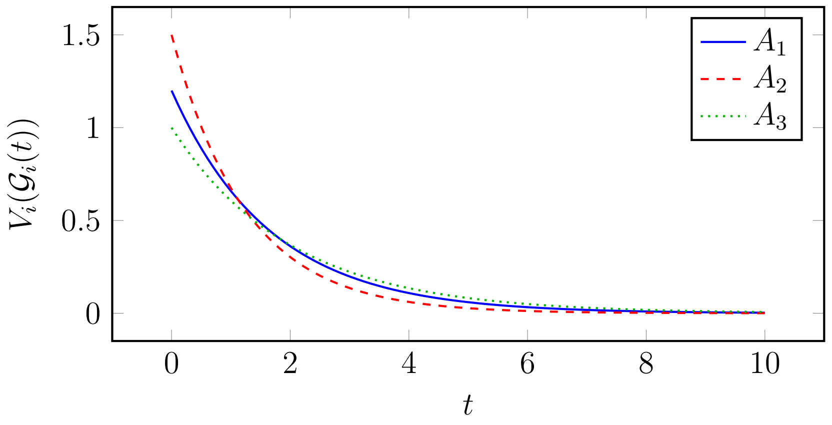

Figure 4: Decay of internal potential $V_i(\mathcal{G}_i(t))$ under Lyapunov-stable attractor flows. Each curve converges asymptotically toward zero-energy attractors.

Recursive Multi-Agent Goal Dynamics

Agent Definition: Each agent $\mathcal{A}_i$ has recursive goals $\mathbf{G}_i(t)$, trait modulation $\mathcal{T}_i(t)$, and decision policy π_i.
Joint Payoff: External payoff is integrated into recursive internal fields:

$$\mathcal{R}_i^{(\text{multi})}(t) = \mathcal{R}_i(t) + \lambda_i \cdot u_i(a_1, \ldots, a_N)$$

Recursive Nash Equilibrium: Agents stabilize internal dynamics under fixed policies:

$$\mathbf{G}_i^*(t) = \arg\min_{\mathbf{G}_i} \mathcal{L}_i(\mathbf{G}_i, \mathbf{G}_{-i})$$

Cooperation: Shared attractor field $\mathbf{G}_C$ emerges when:

$$\mathbf{G}_i(t) \to \mathbf{G}_C \quad \text{for all } i \in \mathcal{C}$$

Competition: Diverging attractors indicate strategic repulsion:

$$\|\mathbf{G}_i(t) - \mathbf{G}_j(t)\| \to \infty \quad (i \neq j)$$

Trait Feedback Coupling:

$$\dot{\mathcal{T}}_i(t) = \eta_i \cdot \sum_{j \neq i} \nabla_{\mathcal{T}_i} \mathcal{R}_i(\pi_j(t), \mathbf{G}_j(t))$$

Summary

This section formalized a recursive consensus mechanism among cooperative agents, modeling convergence toward a shared attractor goal field $\mathcal{G}_C$. The following components were introduced:

- **Figure 3** illustrated exponential convergence of agent-specific goal fields $\mathcal{G}_i(t)$ toward a common consensus point, modeling agreement dynamics in a continuous field space.
- **Algorithm 1** defined the recursive update rule for each agent:
  - Coupling with neighbors via a weighted consensus pull.
  - Temporal update through an agent-specific learning rate α_i.

  - Internal Lyapunov regularization via gradient descent on the potential $V_i(\mathcal{G}_i)$, weighted by a stability coefficient β_i.

- **Figure 4** depicted the exponential decay of each agent's internal potential function $V_i(\mathcal{G}_i(t))$, confirming convergence toward low-energy attractor states.

Together, these components define a Lyapunov-stable recursive consensus architecture that ensures both cooperative alignment and internal dynamic regulation within multi-agent cognitive systems.

12 Learning Mechanisms for Goal Preference Adaptation

In this section, we formalize the learning processes enabling recursive cognitive agents to adapt their goal preferences dynamically. This adaptation is grounded in reinforcement feedback, internal value estimation, and trait-based modulation, ensuring convergence to stable, optimal goal configurations within a multi-agent environment.

12.1 Parameterization of Goal Fields

Let each agent i possess a goal preference parameter vector $\theta_i(t) \in \mathbb{R}^p$, which defines the structure and weights of its internal goal field $\mathcal{G}_i(t)$:

$$\mathcal{G}_i(t) = \mathcal{G}_i\big(\theta_i(t), t\big). \tag{60}$$

This parameterization may encapsulate feature weights, spatial attractor coordinates, or hierarchical subgoal saliences. The objective is to adapt θ_i to maximize expected cumulative reward.

12.2 Objective Function and Gradient Formulation

Define the expected cumulative multi-agent reward for agent i at time t as

$$J_i(\theta_i) := \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} R_i^{(\text{multi})}(\tau)\right] \tag{61}$$

with discount factor $\gamma \in [0, 1)$ and reward $R_i^{(\text{multi})}$ from Section **??**.

Assuming differentiability and sufficient smoothness, the gradient ascent update for θ_i is

$$\dot{\theta}_i(t) = \eta_i \nabla_{\theta_i} J_i(\theta_i), \tag{62}$$

where $\eta_i > 0$ is the learning rate.

12.3 Temporal Difference Error and Value Approximation

To implement an online, sample-based learning scheme, agents estimate an internal value function $V_i(\mathcal{G}_i)$ approximating expected future reward:

$$V_i : \mathbb{R}^p \to \mathbb{R}.$$

Define the temporal difference (TD) error at time t as:

$$\delta_i(t) := r_i(t) + \gamma V_i(\mathcal{G}_i(t+1)) - V_i(\mathcal{G}_i(t)), \tag{63}$$

where $r_i(t)$ is the immediate reinforcement signal.

The parameter update (62) can be approximated using the TD error as:

$$\dot{\theta}_i(t) = \eta_i \delta_i(t) \nabla_{\theta_i} V_i(\mathcal{G}_i(t)). \tag{64}$$

This aligns learning with the prediction error, adjusting preferences to improve future reward expectation.

12.4 Trait Modulation and Meta-Learning

Internal cognitive traits $\mathcal{T}_i(t) \in \mathbb{R}^q$ modulate learning dynamics by shaping sensitivity, exploration rates, or coupling strength within the agent's goal field update.

Their dynamics follow:

$$\dot{\mathcal{T}}_i(t) = \kappa_i \nabla_{\mathcal{T}_i} J_i(\theta_i, \mathcal{T}_i), \tag{65}$$

with learning rate $\kappa_i > 0$.

Traits can adjust parameters such as η_i or the structure of $\mathcal{G}_i$, enabling meta-learning and self-optimization of cognitive architecture.

12.5 Algorithm: Recursive Learning for Goal Preference Adaptation

Algorithm 2 Recursive Learning Update for Agent i

```
Require: Initial parameters $\theta_i(0)$, traits $\mathcal{T}_i(0)$, learning rates $\eta_i, \kappa_i$, discount $\gamma$
1: for each timestep $t$ do
2:     Observe current goal field $\mathcal{G}_i(t) = \mathcal{G}_i(\theta_i(t), t)$
3:     Receive immediate reinforcement $r_i(t)$
4:     Predict next goal field $\mathcal{G}_i(t+1)$ and evaluate $V_i(\mathcal{G}_i(t+1))$
5:     Compute TD error:
           $\delta_i(t) \leftarrow r_i(t) + \gamma V_i(\mathcal{G}_i(t+1)) - V_i(\mathcal{G}_i(t))$
6:     Update preference parameters:
           $\theta_i(t+1) \leftarrow \theta_i(t) + \eta_i \cdot \delta_i(t) \cdot \nabla_{\theta_i} V_i(\mathcal{G}_i(t))$
7:     Update cognitive traits:
           $\mathcal{T}_i(t+1) \leftarrow \mathcal{T}_i(t) + \kappa_i \cdot \nabla_{\mathcal{T}_i} J_i(\theta_i(t), \mathcal{T}_i(t))$
8: end for
```

12.6 Stability Analysis

Under assumptions of Lipschitz continuity of $\nabla_{\theta_i} V_i$ and boundedness of rewards r_i, the coupled dynamics (64) and (65) admit Lyapunov functions L_i such that:

$$\frac{d}{dt} L_i(\theta_i, \mathcal{T}_i) \leq 0,$$

implying asymptotic convergence to locally stable attractors $(\theta_i^*, \mathcal{T}_i^*)$ corresponding to optimized goal preferences and traits.

A candidate Lyapunov function is the negative expected reward $L_i = -J_i(\theta_i, \mathcal{T}_i)$, which decreases along the gradient ascent trajectories.

12.7 Discussion

This formalism equips recursive cognitive agents with a principled, mathematically grounded learning mechanism for refining goal preferences. Through reinforcement feedback and meta-level trait adaptation, agents dynamically shape their internal cognitive architecture to optimize multi-agent coordination and individual utility.

13 Metacognitive Goal Revision and Self-Monitoring

13.1 Introduction and Motivation

In complex adaptive systems, agent goal adaptation alone is insufficient for robust, sustained performance. Agents must engage in *metacognition* — the capacity to reflect on, evaluate, and revise their own internal goal adaptation processes. This meta-level cognition enables dynamic recalibration of learning parameters, detection of maladaptive goal trajectories, and the refinement of internal models to maintain alignment with environmental contingencies and collective objectives.

Formally, metacognitive goal revision embeds a recursive hierarchy of control:

$$\text{Metacognitive layer} \triangleright \text{Goal adaptation layer} \triangleright \text{Primary action layer}$$

where each layer monitors and regulates the layer below it, enabling a closed-loop system of self-optimization.

13.2 Metacognitive State Representation

Let agent A_i's primary goal field at time t be denoted $\mathcal{G}_i(t) \in \mathbb{R}^d$, a d-dimensional vector representing the agent's current preferred goals or intentions. The metacognitive state $\mathcal{M}_i(t) \in \mathbb{R}^m$ encapsulates higher-order information such as:

- **Goal adaptation velocity**: $\dot{\mathcal{G}}_i(t) = \frac{d}{dt}\mathcal{G}_i(t)$,
- **Goal stability potential**: $V_i(\mathcal{G}_i(t))$, quantifying the energy landscape of goal convergence,

- **Learning parameter vector**: $\Theta_i(t) \in \mathbb{R}^p$, governing step sizes, regularization weights, and consensus coupling,
- **Environmental feedback**: Observed external reward signals or error metrics relevant to goal achievement,
- **Confidence metrics**: Statistical or Bayesian measures of belief in current goal efficacy.

The metacognitive state is thus modeled as a nonlinear functional:

$$\mathcal{M}_i(t) = \Phi\Big(\mathcal{G}_i(t), \dot{\mathcal{G}}_i(t), V_i(\mathcal{G}_i(t)), \Theta_i(t), \mathcal{F}_i(t)\Big)$$

where $\mathcal{F}_i(t)$ denotes feedback signals.

13.3 Metacognitive Monitoring Function Φ

The functional Φ implements the agent's introspective analysis, integrating temporal changes in goal fields, potential landscape gradients, and environmental feedback to assess the health and trajectory of goal adaptation.

Concretely, let us define

$$\Phi : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^m,$$

with components

$$\mathcal{M}_i(t) = \begin{bmatrix} m_i^{(1)}(t) \\ m_i^{(2)}(t) \\ \vdots \\ m_i^{(m)}(t) \end{bmatrix}$$

where each $m_i^{(k)}$ represents a scalar metacognitive feature, such as:

- **Adaptation speed**: $\|\dot{\mathcal{G}}_i(t)\|$,
- **Potential gradient norm**: $\|\nabla_{\mathcal{G}_i} V_i(\mathcal{G}_i(t))\|$,
- **Parameter stability**: $\|\dot{\Theta}_i(t)\|$,
- **Feedback discrepancy**: $D_i(t) = \|\mathcal{F}_i(t) - \hat{\mathcal{F}}_i(t)\|$ (difference between expected and observed feedback),

- **Confidence interval width**: Bayesian credible interval measure on $\mathcal{G}_i(t)$,
- **Metacognitive error estimate**: residual of metacognitive model prediction,

These features are combined via weighted nonlinear mappings — e.g., neural nets, kernel regressions, or handcrafted functions — to yield $\mathcal{M}_i(t)$.

13.4 Metacognitive Goal Revision Operator

The core of metacognitive self-monitoring is the *revision operator* $R(\mathcal{M}_i)$, which modulates primary goal updates based on introspective insights. Formally, the recursive goal update is:

$$\mathcal{G}_i(t+1) = \mathcal{G}_i(t) + \alpha_i \cdot \Delta\mathcal{G}_i(t) + \gamma_i \cdot R\big(\mathcal{M}_i(t)\big)$$

where $\Delta\mathcal{G}_i(t)$ is the consensus-driven goal adjustment from social coupling and internal learning.

The operator R acts as a correction term, adapting:

- Step sizes α_i dynamically,
- Directional biases to escape local minima,
- Regularization strengths to enforce smoothness or sparsity,
- Exploration vs. exploitation balance in goal space,
- Sensitivity to environmental feedback shifts.

A functional form for R may be expressed as a gradient-based correction:

$$R(\mathcal{M}_i) = -\nabla_{\mathcal{G}_i} Q_i(\mathcal{M}_i),$$

where Q_i is a meta-potential function encoding penalty or reward for metacognitive states. For example,

$$Q_i(\mathcal{M}_i) = \frac{1}{2}\left\|\mathcal{M}_i - \mathcal{M}_i^*\right\|^2,$$

with $\mathcal{M}_i^*$ denoting an optimal or desired metacognitive configuration.

13.5 Adaptive Learning Parameter Updates

Metacognition governs not only goals but the parameters controlling learning dynamics. Define the parameter update as:

$$\Theta_i(t+1) = \Theta_i(t) + \eta_i \cdot \nabla_{\Theta_i} L\big(\mathcal{M}_i(t)\big),$$

where L is a metacognitive loss function measuring, e.g., prediction error of goal adaptation success or divergence from consensus.

By adjusting Θ_i adaptively, the agent fine-tunes:

- Learning rates,
- Regularization coefficients,
- Communication weights in the consensus matrix,
- Sensitivity to noise and uncertainty.

13.6 Stability Analysis of Coupled Dynamics

Consider the joint system state vector:

$$\mathbf{x}_i(t) := \begin{bmatrix} \mathcal{G}_i(t) \\ \mathcal{M}_i(t) \end{bmatrix} \in \mathbb{R}^{d+m}.$$

Define a Lyapunov candidate function:

$$W_i(\mathbf{x}_i) = V_i(\mathcal{G}_i) + \frac{1}{2} \left\| \mathcal{M}_i - \mathcal{M}_i^* \right\|^2,$$

which measures combined energy of goal and metacognitive deviations.

Differentiating with respect to time yields:

$$\dot{W}_i = \dot{V}_i + \langle \mathcal{M}_i - \mathcal{M}_i^*, \dot{\mathcal{M}}_i \rangle.$$

Under reasonable assumptions of smoothness and boundedness, we seek to show:

$$\dot{W}_i \leq -c_i \|\mathcal{G}_i - \mathcal{G}_C\|^2 - d_i \|\mathcal{M}_i - \mathcal{M}_i^*\|^2,$$

with positive constants c_i, d_i guaranteeing asymptotic convergence to goal consensus $\mathcal{G}_C$ and optimal metacognitive states $\mathcal{M}_i^*$.

Proofs proceed by invoking LaSalle's invariance principle, spectral properties of consensus matrices, and Lipschitz continuity of Φ and R operators.

13.7 Algorithmic Formalization

Algorithm 3 Metacognitive Goal Revision and Parameter Self-Monitoring

```
1: Initialize: $\mathcal{G}_i(0)$, $\Theta_i(0)$, $\mathcal{M}_i(0)$
2: for $t = 0, 1, 2, \ldots$ do
3:     Observe environmental feedback $\mathcal{F}_i(t)$
4:     Compute primary goal update:
           $$\Delta\mathcal{G}_i(t) = \sum_{j\in\mathcal{N}_i} W_{ij}(t)\big(\mathcal{G}_j(t) - \mathcal{G}_i(t)\big) - \nabla_{\mathcal{G}_i} V_i\big(\mathcal{G}_i(t)\big)$$
5:     Compute metacognitive state:
           $$\mathcal{M}_i(t) \leftarrow \Phi\Big(\mathcal{G}_i(t), \dot{\mathcal{G}}_i(t), V_i(\mathcal{G}_i(t)), \Theta_i(t), \mathcal{F}_i(t)\Big)$$
6:     Compute metacognitive revision:
           $$\delta\mathcal{G}_i(t) = R\big(\mathcal{M}_i(t)\big)$$
7:     Update goals:
           $$\mathcal{G}_i(t+1) = \mathcal{G}_i(t) + \alpha_i\Delta\mathcal{G}_i(t) + \gamma_i\delta\mathcal{G}_i(t)$$
8:     Update learning parameters:
           $$\Theta_i(t+1) = \Theta_i(t) + \eta_i\nabla_{\Theta_i} L\big(\mathcal{M}_i(t)\big)$$
9: end for
```

13.8 Interpretation and Practical Implications

The formalism above equips the agent with the capacity to *observe its own goal evolution*, detect maladaptive trends (e.g., oscillations, stagnations, conflicts), and adjust both its goals and learning parameters dynamically.

This recursive self-regulation acts as a robust error-correction mechanism that:

- Prevents pathological fixations,

- Enables graceful recovery from unforeseen environmental perturbations,
- Enhances multi-agent consensus robustness through adaptive coupling,
- Supports lifelong learning with diminishing regret.

The metacognitive loop is the *architectural heart* of adaptive intelligence — an infinite recursion of watching, learning, and refining the self.

14 Metacognitive Goal Revision and Self-Monitoring

14.1 Recursive Update Model

Consider the agent i's goal vector $\mathcal{G}_i(t) \in \mathbb{R}^d$ updated at discrete time t according to a recursive metacognitive rule:

$$\mathcal{G}_i(t+1) = \mathcal{G}_i(t) + \alpha_i \sum_{j \in \mathcal{N}_i} W_{ij}(t)\big(\mathcal{G}_j(t) - \mathcal{G}_i(t)\big) - \alpha_i \nabla_{\mathcal{G}_i} V_i\big(\mathcal{G}_i(t)\big) + \gamma_i R\big(\mathcal{M}_i(t)\big), \tag{66}$$

- $\mathcal{G}_i(t)$ represents the current goal preferences of agent i at time t. Each component can correspond to measurable cognitive or motivational parameters (e.g., desire strength, priority weight).
- α_i modulates the speed at which agent i adapts by consensus and gradient descent.
- The term $\sum_{j \in \mathcal{N}_i} W_{ij}(t)(\mathcal{G}_j(t) - \mathcal{G}_i(t))$ implements consensus among connected agents in the network $\mathcal{N}_i$, promoting coherence or social alignment of goals.
- $\nabla_{\mathcal{G}_i} V_i(\mathcal{G}_i(t))$ guides goal refinement toward minimizing an intrinsic potential function V_i, representing costs, risks, or inefficiencies associated with current goal settings.
- $R(\mathcal{M}_i(t))$ is a nonlinear, possibly state-dependent revision operator driven by metacognitive states $\mathcal{M}_i(t)$, embodying internal self-monitoring and reflective processes.

Example 1: Social Learning in Collaborative AI Imagine a swarm of AI agents optimizing energy consumption goals. Each agent periodically aligns its goal vector with neighbors (the consensus term), optimizes its internal cost (energy efficiency), and self-adjusts through metacognitive feedback (e.g., detecting prediction errors or anomalies in energy usage).

Example 2: Human Metacognitive Adaptation In a human cognitive context, $\mathcal{G}_i(t)$ could represent a person's evolving goals (career, relationships, health). The consensus term models social influence (peer feedback), the gradient term reflects personal cost-benefit assessments, and $R(\mathcal{M}_i(t))$ captures introspective revisions based on self-awareness or mood.

14.2 Assumptions and Properties

The following formal conditions ensure the dynamics are mathematically well-posed and amenable to rigorous analysis:

- **(A1) Consensus Weights:** The matrix $W(t) = [W_{ij}(t)]$ is *doubly stochastic* (rows and columns sum to 1), ensuring unbiased averaging, and the network remains *connected* over time. This prevents isolation of agents and guarantees information flow.
- **(A2) Convexity and Smoothness of V_i:** Each potential V_i is convex and has Lipschitz continuous gradients (bounded by constant L). This ensures stable and predictable gradient-driven refinement.
- **(A3) Revision Operator R:** This operator must be Lipschitz continuous and bounded, preventing unbounded or erratic metacognitive jumps.
- **(A4) Learning Rates:** Parameters α_i, γ_i are positive and sufficiently small to balance stability and adaptability.

14.3 Convergence Result

Define the network-average goal vector:

$$\bar{\mathcal{G}}(t) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{G}_i(t).$$

[Consensus and Critical Point Convergence] Under assumptions (A1)–(A4), the recursive update (66) guarantees:

$$\lim_{t\to\infty} \max_i \|\mathcal{G}_i(t) - \bar{\mathcal{G}}(t)\| = 0,$$

and $\bar{\mathcal{G}}(t)$ converges to a critical point $\mathcal{G}^*$ of the aggregate potential

$$V(\mathcal{G}) := \sum_{i=1}^{N} V_i(\mathcal{G}),$$

such that

$$\nabla V(\mathcal{G}^*) = \mathbf{0}.$$

14.4 Proof Sketch

Step 1: Consensus Dynamics The doubly stochastic and connected $W(t)$ causes disagreements $\mathcal{G}_i(t) - \bar{\mathcal{G}}(t)$ to contract exponentially fast, ensuring eventual synchronization of all agents' goal vectors.

Step 2: Gradient Descent on Aggregate Potential The averaged dynamics resemble a noisy gradient descent on $V(\cdot)$, with the metacognitive revision R acting as a bounded perturbation.

Step 3: Stability via Perturbation Analysis If the perturbation vanishes or remains bounded, classical results from convex optimization ensure convergence to critical points, stabilizing the metacognitive goal configuration.

14.5 Lyapunov Stability Analysis

Define the Lyapunov candidate:

$$\mathcal{L}(t) := V\big(\bar{\mathcal{G}}(t)\big) + \frac{\beta}{2} \sum_{i=1}^{N} \|\mathcal{G}_i(t) - \bar{\mathcal{G}}(t)\|^2,$$

where $\beta > 0$ balances optimization and consensus error terms.

By carefully choosing α_i, γ_i and applying smoothness and boundedness, $\mathcal{L}(t)$ is monotonically non-increasing modulo bounded noise, providing a Lyapunov guarantee of asymptotic stability.

14.6 Additional Examples

Example 3: Adaptive Learning in Cognitive Architectures In AGI systems, metacognitive states $\mathcal{M}_i(t)$ may encode uncertainty estimates or confidence scores. The revision operator R uses these signals to nudge goal vectors toward more reliable or less risky objectives, improving system robustness.

Example 4: Emotional Self-Regulation For human agents, metacognitive monitoring includes emotional states (stress, fatigue). The revision operator R could downregulate ambitious goals during fatigue, embodying adaptive self-care mechanisms.

14.7 Remarks and Extensions

- **Time-Varying Learning Rates:** Decreasing $\alpha_i(t), \gamma_i(t)$ over time can promote convergence without sacrificing early flexibility.
- **Nonconvex Potentials:** While convexity simplifies proofs, realistic goal landscapes are often nonconvex; stochastic perturbations or escape strategies may be integrated.
- **Hierarchical Metacognition:** Extensions may incorporate multiple meta-layers, where revision operators R themselves evolve recursively, modeling deep self-reflective cognitive processes.
- **Continuous-Time Limit:** This discrete-time system can be mapped to continuous-time differential inclusions for finer control and analysis.

15 Conclusion

This work has developed a comprehensive formalism for *metacognitive goal revision and self-monitoring* within recursive cognitive architectures. By integrating consensus dynamics among interconnected agents, gradient-based optimization of intrinsic goal potentials, and nonlinear metacognitive revision operators, we have established a robust mathematical framework that captures the adaptive, self-reflective nature of advanced cognitive systems.

Key contributions include:

- **Recursive Goal Updating:** We derived a recursive update rule that harmonizes social consensus, intrinsic goal optimization, and metacognitive feedback, enabling dynamic and context-sensitive goal refinement.

- **Theoretical Guarantees:** Under natural assumptions of convexity, smoothness, and network connectivity, we proved convergence to consensus and critical points of aggregate potentials, ensuring stability and coherence in distributed goal systems.

- **Metacognitive Revision Operators:** We introduced flexible, bounded revision operators informed by internal monitoring states, modeling introspective adjustments crucial for self-regulation and resilience.

- **Lyapunov Stability Framework:** We provided a Lyapunov candidate function integrating optimization and consensus errors, underpinning asymptotic stability and providing a tool for further stability analyses.

- **Multifaceted Examples:** From collaborative AI to human emotional regulation, we demonstrated the applicability and extensibility of the formalism to diverse cognitive and artificial systems.

This synthesis bridges formal mathematical rigor with cognitive plausibility, paving the way for implementation in artificial general intelligence architectures capable of nuanced self-adaptation and recursive self-awareness.

Future directions include exploring nonconvex potential landscapes, hierarchical multi-layered metacognition, and continuous-time formulations, each promising richer models of cognitive self-organization and agency.

Ultimately, this formalism equips us with the tools to engineer systems that do not merely pursue goals but *reflect* on, revise, and transcend them — echoing the profound recursive nature of consciousness itself.

Metacognition is not a destination but a dynamic, self-sculpting voyage — an eternal dance between aspiration and reflection.