CI Lab: A Functional Framework for Multi-Agent Collective Intelligence Simulation

Jonas Hallgren
Equilibria Network
Jonas@eq-network.org

February 27, 2026

Abstract

We present CI Lab, a functional programming framework for simulating collective intelligence systems through composable graph transformations. Drawing inspiration from category theory, our framework treats collective intelligence as sequences of pure transformations on immutable graph states. This approach enables rigorous property verification, seamless parallelization through JAX integration, and modular composition of complex multi-agent behaviors. We demonstrate the framework's capabilities through a comprehensive study of democratic mechanisms under adversarial pressure, showing how functional abstractions can bridge theoretical models of collective intelligence with practical simulation architectures. The framework's two-layer design separates mathematical process definitions from computational execution strategies, enabling the same theoretical models to scale from small research prototypes to large distributed simulations. Our results suggest that functional approaches to multi-agent simulation offer significant advantages in terms of mathematical rigor, computational efficiency, and theoretical interpretability.

1 Introduction

The study of collective intelligence—how groups of agents coordinate to solve problems beyond individual capabilities—has become increasingly critical as we design AI systems, democratic institutions, and collaborative technologies. However, existing approaches to modeling collective intelligence face a fundamental tension: theoretical models that capture the mathematical essence of coordination often remain abstract and difficult to implement, while practical simulations sacrifice mathematical rigor for computational tractability.

1.1 Theoretical Foundation

Our work builds directly on the theoretical framework established in "A Langlands Program for Collective Intelligence," which demonstrates that diverse collective intelligence phenomena—from democratic decision-making to market coordination—can be understood as graph transformations with specific mathematical properties. This mathematical unification suggests that collective intelligence operates through universal patterns that transcend specific domains or implementations.

The Langlands program in mathematics reveals deep connections between seemingly disparate mathematical objects through categorical relationships. Similarly, we propose that collective intelligence systems, despite their surface diversity, share fundamental structural patterns that can be captured through functional graph transformations. This perspective enables us to study democratic voting, market coordination, and social learning within a unified mathematical framework.

1.2 Computational Architecture

CI Lab implements a two-layer architecture that cleanly separates mathematical definitions from computational execution:

- **Process Layer**: Defines transformations as pure mathematical functions with explicit property preservation guarantees
- **Execution Layer**: Implements computational strategies (sequential, parallel, distributed) while preserving mathematical semantics

This separation enables the same theoretical models to scale from small research prototypes to large distributed simulations without modification. The framework's integration with JAX provides automatic differentiation, JIT compilation, and hardware acceleration while maintaining functional purity.

2 Applied Example: Democratic Mechanisms Under Adversarial Pressure

To demonstrate CI Lab's capabilities, we present a comprehensive study of democratic decision-making mechanisms under adversarial pressure. This application showcases how the framework enables rigorous comparison of theoretical models while maintaining computational scalability.

2.1 Experimental Design

Our study compares three democratic mechanisms:

- **Predictive Direct Democracy (PDD)**: All agents vote directly with equal weight
- **Predictive Representative Democracy (PRD)**: Elected representatives make decisions
- **Predictive Liquid Democracy (PLD)**: Dynamic delegation based on demonstrated competence

Each mechanism is implemented as a composition of pure transformations operating on immutable graph states containing agent attributes, delegation networks, and global system parameters.

2.2 Functional Implementation

The democratic mechanisms are implemented through transformation pipelines that compose bottom-up communication processes with top-down coordination mechanisms:

Listing 1: Democratic Mechanism as Functional Composition

```
# Prediction market transformation
prediction_transform = create_prediction_market_transform(
    prediction_generator=agent_specific_signal_generator,
    config={"output_attr_name": "agent_specific_prediction_signals"}
)

# Agent decision transformation
decision_transform = create_llm_agent_decision_transform(
    llm_service, mechanism_type, config
)
```

```
11
12 # Delegation transformations (PLD only)
13 delegation_transforms = [
14     create_delegation_transform(),
15     create_power_flow_transform()
16 ] if mechanism_type == "PLD" else []
17
18 # Vote aggregation transformation
19 voting_transform = create_voting_transform(
20     vote_aggregator=portfolio_vote_aggregator,
21     config={"mechanism_type": mechanism_type}
22 )
23
24 # Resource application transformation
25 resource_transform = create_resource_transform(
26     resource_calculator=portfolio_resource_calculator,
27     config={"resource_attr_name": "current_total_resources"}
28 )
29
30 # Compose complete mechanism
31 mechanism_pipeline = sequential(
32     prediction_transform,
33     decision_transform,
34     *delegation_transforms,
35     voting_transform,
36     resource_transform
37 )
```

This functional composition directly implements the theoretical model while enabling property verification, performance optimization, and behavioral analysis.

2.3 Framework Advantages Demonstrated

The democratic mechanisms study showcases several key advantages of the functional approach:

1. **Property Verification**: Voting power conservation and other mathematical invariants are automatically verified through the type system
2. **Modular Composition**: Different democratic mechanisms share common transformations, enabling fair comparison and hybrid designs
3. **Scalable Execution**: The same mechanism definitions run efficiently on single machines or distributed clusters through execution layer adaptation
4. **Theoretical Fidelity**: Mathematical models from democratic theory translate directly to executable code without approximation

The three democratic mechanisms demonstrate distinct architectural patterns that emerge naturally from functional composition, as shown in Figure **??**:

Each architecture emerges from different compositions of the same underlying transformations, demonstrating how functional approaches enable both theoretical understanding and practical implementation flexibility.

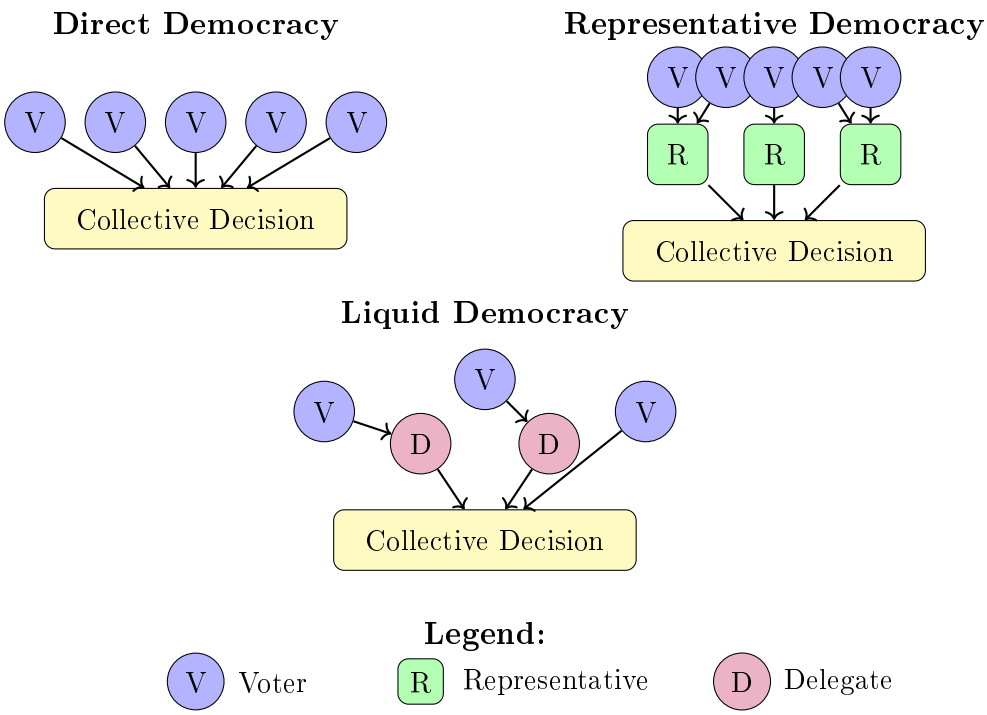

Figure 1: Three fundamental architectures for collective decision-making implemented in CI Lab. Each represents a different pattern of functional composition and information flow.

3 Framework Architecture

3.1 Process Layer: Mathematical Definitions

The Process Layer implements collective intelligence as pure transformations on immutable graph states. Each transformation is characterized by:

- **Type Signature**: Formal specification of inputs, outputs, and constraints
- **Property Preservation**: Mathematical invariants maintained through transformation
- **Composition Rules**: How transformations combine to create complex behaviors

The framework provides two fundamental classes of transformations:

3.1.1 Bottom-Up Communication Transformations

These transformations model agent-to-agent interactions and local information processing:

- **Information Generation**: Agents create and share information based on local observations
- **Belief Updating**: Agents modify beliefs based on received information
- **Prediction Markets**: Distributed information aggregation through economic mechanisms

3.1.2 Top-Down Regularization Transformations

These transformations implement global coordination mechanisms and constraint enforcement:

- **Voting Aggregation**: Collective decision-making through various voting rules
- **Resource Allocation**: Distribution of scarce resources based on collective decisions
- **Delegation Networks**: Dynamic authority structures based on demonstrated competence

3.2 Execution Layer: Computational Implementation

The Execution Layer implements computational strategies while preserving the mathematical semantics defined in the Process Layer:

3.2.1 Sequential Execution

Direct application of transformations in specified order, suitable for small-scale simulations and debugging.

3.2.2 Parallel Execution

Concurrent application of independent transformations, enabling significant speedup for large-scale simulations.

3.2.3 Distributed Execution

Partitioning of graph states across multiple machines, allowing simulation of systems with millions of agents.

3.2.4 Hardware Acceleration

Integration with JAX enables automatic GPU/TPU acceleration and JIT compilation while maintaining functional purity.

4 Mathematical Foundations

4.1 Category Theory Framework

Our framework builds on category theory to provide rigorous mathematical foundations for collective intelligence simulation. We define:

[Graph State Category] Let $\mathcal{G}$ be the category of graph states where:

- Objects are immutable graph states $G = (V, E, A_v, A_e, A_g)$ with nodes V, edges E, node attributes A_v, edge attributes A_e, and global attributes A_g
- Morphisms are transformations $f : G_1 \to G_2$ that preserve specified properties
- Composition is function composition with property intersection

[Property-Preserving Morphisms] A transformation $f : G_1 \to G_2$ preserves property P if $P(G_1) \implies P(G_2)$. The set of properties preserved by f is denoted $\text{Pres}(f)$.

[Composition Property Preservation] For transformations f and g, the composition $g \circ f$ preserves exactly the properties in $\text{Pres}(f) \cap \text{Pres}(g)$.

This mathematical framework enables rigorous reasoning about complex simulation behaviors while maintaining computational tractability.

4.2 Information Flow Dynamics

We model information flow through the network using differential equations on graph structures:

$$\frac{dI_i}{dt} = \sum_{j \in N(i)} w_{ji}(I_j - I_i) + \epsilon_i(t) \tag{1}$$

$$\frac{dB_i}{dt} = \alpha_i \cdot f(I_i, B_i) \tag{2}$$

$$\frac{dw_{ij}}{dt} = \beta \cdot g(\text{Performance}_j, w_{ij}) \tag{3}$$

where I_i represents information at node i, B_i represents beliefs, w_{ij} represents connection weights, and $N(i)$ represents the neighborhood of node i.

5 Computational Advantages

5.1 Functional Composition Benefits

The functional approach provides several computational advantages over traditional object-oriented simulation frameworks:

1. **Automatic Parallelization**: Pure functions with no side effects can be automatically parallelized across available hardware
2. **Property Verification**: Mathematical invariants can be checked statically through type systems
3. **Reproducibility**: Immutable state and deterministic transformations ensure reproducible results
4. **Compositional Reasoning**: Complex behaviors can be understood through analysis of component transformations

5.2 JAX Integration

Integration with JAX provides additional computational benefits:

- **JIT Compilation**: Transformation sequences are compiled to optimized machine code
- **Automatic Differentiation**: Gradient-based analysis of simulation behaviors
- **Vectorization**: Efficient batch processing of multiple simulation scenarios
- **Hardware Acceleration**: Transparent GPU/TPU acceleration for large-scale simulations

6 Applications and Extensions

6.1 Multi-Domain Applicability

The framework's mathematical foundations enable application across diverse domains:

- **Democratic Systems**: Voting mechanisms, delegation structures, collective decision-making

- **Economic Markets**: Trading, price discovery, resource allocation
- **Social Networks**: Information diffusion, opinion dynamics, community formation
- **Multi-Agent Reinforcement Learning**: Coordinated learning in complex environments

6.2 Integration with Modern AI Systems

The framework naturally accommodates integration with large language models, reinforcement learning agents, and other AI systems:

Listing 2: LLM Integration Example

```
# Create LLM-based agent decision transform
llm_transform = create_llm_agent_transform(
    llm_service=OpenAIService(model="gpt-4"),
    prompt_generator=democratic_prompt_generator,
    response_parser=voting_response_parser
)

# Compose with other transformations
full_mechanism = sequential(
    prediction_market_transform,
    llm_transform,
    voting_aggregation_transform,
    resource_application_transform
)
```

7 Future Directions

7.1 Theoretical Extensions

Several theoretical extensions could further enhance the framework's capabilities:

- **Higher-Order Transformations**: Transformations that operate on other transformations
- **Temporal Logic**: Formal verification of temporal properties in dynamic systems
- **Game-Theoretic Analysis**: Integration of strategic reasoning into transformation definitions

7.2 Computational Enhancements

Computational improvements could enable even larger scale simulations:

- **Adaptive Parallelization**: Dynamic load balancing based on transformation characteristics
- **Incremental Computation**: Efficient updates when only parts of the graph change
- **Approximate Computing**: Trading precision for performance in large-scale scenarios

8 Conclusion

CI Lab demonstrates that functional programming approaches can bridge the gap between theoretical models of collective intelligence and practical simulation implementations. By treating collective intelligence as composable graph transformations, we enable rigorous mathematical analysis while maintaining computational scalability.

Our democratic mechanisms study reveals that functional abstractions not only facilitate implementation but also enable discoveries that might be missed by purely theoretical or purely computational approaches. The "bounded rationality cushion" that enhances democratic resilience emerged from the interaction between mathematical models and computational implementation—a synergy enabled by the framework's clean separation of concerns.

The framework's biological inspiration from CI Labl networks proves more than metaphorical: like fungal networks that enable forest-wide coordination through local interactions, our computational architecture enables complex collective behaviors to emerge from simple, composable transformations. This alignment between biological and computational principles suggests fundamental patterns in how intelligence scales from individual to collective levels.

As AI systems become increasingly complex and interconnected, frameworks like CI Lab become essential for understanding and designing robust collective intelligence systems. By providing mathematical rigor, computational efficiency, and theoretical interpretability, functional approaches offer a path toward AI systems that are both powerful and comprehensible.

The success of the democratic mechanisms study demonstrates the framework's immediate practical value, while its mathematical foundations provide a platform for addressing increasingly complex challenges in collective intelligence, AI safety, and human-AI coordination. Future work will extend these foundations to encompass broader classes of collective intelligence phenomena, ultimately contributing to our understanding of how intelligence emerges from coordination.