Detailed Roadmap for the Modular Proof Framework

Research Team

May 23, 2025

Abstract

This document presents a comprehensive roadmap for developing a modular proof framework aimed at proving the Riemann Hypothesis (RH) and its generalizations. The framework emphasizes recursive refinement, dynamic prime-zero mapping, and transcendental theory, while leveraging existing tools in number theory, algebraic geometry, automorphic forms, and numerical methods. The roadmap outlines four key phases of development, with detailed tasks, milestones, and references to foundational works.

1 Introduction

The Riemann Hypothesis and its generalizations have remained central problems in mathematics for over a century. This roadmap outlines a step-by-step approach for constructing a modular proof framework that operates across different mathematical domains. The framework is based on the conceptual metaphor of an archer shooting arrows into a fluid-like medium, where agents (arrows) traverse the error space of L-functions under the guidance of a general recursion framework (archer). The ultimate goal is to establish a lattice of transcendental objects connecting zeros of L-functions to equivalent structures across domains.

2 Phases of Development

2.1 Phase 1: Architecture Design

2.1.1 Define the General Framework (Archer)

The general framework, referred to as the archer, oversees the operation of agents across different domains. It manages the initialization of agents, ensures global error minimization, and facilitates communication between agents.

Tasks

- Specify core responsibilities of the archer, including agent initialization and zero synchronization.
- Develop multi-agent communication protocols for zero sharing and error updates.
- Document core components, including APIs for agent management.

Milestone A fully documented general framework with multi-agent communication.

2.1.2 Model the Error Space

The error space is modeled as a dynamic medium governed by PDE-driven error evolution. Agents traverse this space using recursive refinement, guided by local curvature and gradient information.

Tasks

- Define the error evolution PDE and determine boundary and initial conditions.
- Implement numerical solvers using finite difference or finite element methods.
- Validate the error model through numerical experiments.

Milestone A stable error space model with validated boundary conditions.

2.1.3 Design the Lattice of Transcendental Objects

The lattice of transcendental objects forms the backbone of the framework, linking zeros of L-functions to equivalent structures across domains.

Tasks

- Define initial nodes (zeros, primes, critical points) and edges (equivalences).
- Develop algorithms for dynamic lattice updates and consistency checks.
- Visualize the evolving lattice using graph-based tools.

Milestone A dynamically growing lattice with initial nodes and edges.

2.2 Phase 2: Agent Development

2.2.1 Develop Domain-Specific Agents (Arrows)

Agents, referred to as arrows, are domain-specific components responsible for recursive refinement in different mathematical domains. Each agent operates under the guidance of the general framework (archer) and communicates with other agents to ensure global consistency.

Tasks

- Design a focused agent for the Riemann Hypothesis, implementing recursive refinement and dynamic step control.
- Extend the focused agent to handle Dirichlet L-functions, incorporating periodic boundary conditions.
- Develop agents for automorphic forms, ensuring compatibility with motivic L-functions.

Milestone A set of fully functioning domain-specific agents capable of recursive refinement and zero prediction.

2.2.2 Implement Communication and Synchronization

Agents must communicate and synchronize zero positions across different domains to ensure consistency and accelerate convergence.

Tasks

- Develop zero-sharing protocols for agents to exchange zero positions and error data.
- Implement synchronization rules to resolve conflicts and maintain global consistency.
- Validate the communication system through tests on multi-agent zero prediction.

Milestone A robust multi-agent system with synchronized zero positions and validated communication protocols.

2.3 Phase 3: Cross-Domain Generalization

2.3.1 Extend the Lattice to New Domains

To achieve full generalization, the framework must support diverse mathematical domains, including automorphic forms, motivic L-functions, and modular forms. The lattice of transcendental objects will be extended to include additional nodes and edges corresponding to new structures.

Tasks

- Incorporate nodes representing motivic L-functions and automorphic forms, linking them to existing zeros and critical points.
- Add edges corresponding to new equivalences, such as Langlands correspondences and motivic Galois actions.
- Ensure that lattice updates maintain consistency across domains.

Milestone An extended lattice encompassing multiple domains, validated by consistency checks and numerical experiments.

2.3.2 Apply Category Theory for Generalization

Category theory provides a unifying framework for representing relationships between mathematical structures. By defining categories and functors, the framework can formalize mappings between different domains, enabling agents to transfer knowledge across fields.

Tasks

- Define categories representing arithmetic, geometric, and analytic structures.
- Implement functors that map objects between categories, ensuring that mappings are compatible with existing agents.
- Develop natural transformations to handle situations where agents operate in overlapping domains.

Milestone A category-theoretic framework that formalizes cross-domain mappings and allows agents to generalize their strategies.

—

2.4 Phase 4: Testing and Validation

2.4.1 Unit Tests for Agents

Each agent must pass a rigorous set of unit tests to ensure that it functions correctly in its respective domain. These tests cover recursive refinement, zero prediction, and communication with other agents.

Tasks

- Design unit tests for recursive refinement, including step size control and error correction.
- Implement tests for zero sharing and synchronization across multiple agents.
- Develop tests for handling edge cases, such as high curvature regions and incomplete data on zero distributions.

Milestone A validated set of agents that pass all unit tests for their respective domains.

2.4.2 Lemma Validation and Proof Automation

To ensure the correctness of proofs generated by the framework, every proof is decomposed into small testable lemmas. These lemmas are validated through automated testing, ensuring that each step of the proof is correct and consistent with previous results.

Tasks

- Decompose proofs into small, independent lemmas with clear dependency chains.
- Create a dependency graph showing how each lemma leads to larger proof components.
- Implement scripts to automate lemma validation using numerical and symbolic tools.

Milestone A fully automated testing pipeline for lemma validation, with a complete dependency graph for proofs.

2.5 Phase 5: Future Extensions and AGI Integration

2.5.1 Design Placeholder Modules for AGI Components

While the current framework relies on human-guided exploration and existing mathematical knowledge, future extensions can incorporate AGI components for autonomous conjecture generation and proof synthesis.

Tasks

- Develop placeholder modules for conjecture generation using neural-symbolic integration.
- Implement interfaces for integrating future AGI-based proof assistants.
- Design a knowledge base for storing and retrieving mathematical conjectures, proofs, and strategies.

Milestone Placeholder modules for future AGI components, ensuring that the framework remains adaptable and extensible.

2.5.2 Enhance the Interactive Interface for Human-AI Collaboration

The framework should support an interactive interface that allows human mathematicians to guide agents, input conjectures, and explore zero distributions.

Tasks

- Design a user-friendly interface for defining new problems, conjectures, and hypotheses.
- Enable real-time feedback from agents, including intermediate results and error analyses.
- Capture human strategies for conjecture generation and proof refinement, storing them for future use.

Milestone A fully functional interactive interface for human-guided exploration and collaboration with agents.

3 Thirteen Key Works for Reference

1. Prime Numbers and the Riemann Hypothesis by Barry Mazur and William Stein.
2. The Distribution of Prime Numbers by Harold Davenport.
3. On the Distribution of Zeros of the Riemann Zeta Function by H.L. Montgomery.
4. Periods and Transcendental Numbers by Pierre Deligne.
5. The Theory of Motives by Alexander Grothendieck.

6. Transcendental Numbers by Alan Baker.
7. Automorphic Forms and Representations by Daniel Bump.
8. Modular Forms and Fermat's Last Theorem by John Coates and Andrew Wiles.
9. Numerical Recipes: The Art of Scientific Computing by William H. Press et al.
10. Gradient Flows in Metric Spaces and in the Space of Probability Measures by Luigi Ambrosio.
11. PDE and Sobolev Spaces by Robert Adams.
12. Category Theory for the Sciences by David Spivak.
13. Sheaves in Geometry and Logic by Saunders Mac Lane and Ieke Moerdijk.