Title

Proof Sketch Refinement: A Multi-Stage Approach to AI-Assisted Formal Proof Generation

Problem Statement

Current AI-assisted formal proof generation methods often struggle with generating complete, detailed proofs for complex theorems, especially when the proof requires multiple intricate steps or novel insights. This limitation hinders the application of AI in advanced mathematical reasoning and theorem proving.

Motivation

Existing approaches typically focus on generating proofs step-by-step or attempt to produce entire proofs at once, often leading to incomplete or incorrect results for complex theorems. Human mathematicians, on the other hand, often start with a high-level proof sketch and gradually refine it into a formal proof. Mimicking this process could allow AI systems to tackle more complex theorems by breaking down the proof generation into manageable stages. This approach leverages the strengths of large language models in natural language understanding and generation, while also incorporating specialized models for formal reasoning and verification.

Proposed Method

We introduce Proof Sketch Refinement (PSR), a multi-stage approach to AI-assisted formal proof generation. The process consists of four main stages: 1) High-level Sketch Generation: A large language model (LLM) generates a high-level proof sketch in natural language. 2) Intermediate Formalization: A specialized model trained on proof refinement tasks expands and formalizes the sketch into a more detailed proof outline. 3) Formal Proof Generation: Another specialized model converts the detailed outline into a formal proof representation. 4) Verification and Refinement: A formal verification model checks the proof's correctness and suggests modifications if needed. This process iterates until a complete, formally verified proof is obtained or a predefined iteration limit is reached.

Step-by-Step Experiment Plan

Step 1: Dataset Preparation

Collect a diverse set of complex theorems from various mathematical domains, including algebra, analysis, topology, and number theory. Ensure that the dataset includes theorems of varying difficulty levels. Split the dataset into training, validation, and test sets.

Step 2: Model Selection and Fine-tuning

Select appropriate models for each stage of the PSR process. For the high-level sketch generation, use a state-of-the-art LLM such as GPT-4. For the intermediate formalization and formal proof generation stages, fine-tune smaller, specialized models (e.g., T5 or BART) on datasets of mathematical proofs. For the verification stage, use an existing formal proof checker or train a neural proof checker.

Step 3: Implement PSR Pipeline

Develop a pipeline that integrates all four stages of the PSR process. Ensure smooth data flow between stages and implement appropriate error handling and logging mechanisms.

Step 4: Baseline Implementation

Implement state-of-the-art baseline methods for comparison, including direct proof generation and step-by-step proof generation approaches.

Step 5: Evaluation Metrics

Define and implement evaluation metrics, including proof completion rate, correctness (verified by formal proof checkers), and the complexity of theorems successfully proven. Also, implement metrics to assess the quality of intermediate outputs at each stage.

Step 6: Experiment Execution

Run experiments on the test set using both the PSR method and baseline methods. Collect results for all defined metrics.

Step 7: Ablation Studies

Conduct ablation studies to assess the impact of each refinement stage. This includes running experiments with different combinations of stages and varying the number of refinement iterations.

Step 8: Analysis and Visualization

Analyze the results, comparing PSR performance against baselines. Create visualizations to illustrate the refinement process and performance improvements across iterations.

Step 9: Error Analysis

Perform detailed error analysis on cases where PSR fails or underperforms. Categorize error types and identify potential areas for improvement.

Test Case Examples

Baseline Prompt Input (Direct Proof Generation)

Prove that the square root of 2 is irrational.

Baseline Prompt Expected Output (Direct Proof Generation)

Proof: Assume √2 is rational. Then √2 = a/b where a and b are integers with no common factors. Squaring both sides: 2 = a^2/b^2. Multiply both sides by b^2: 2b^2 = a^2. This means a^2 is even, so a must be even. Let a = 2k. Substituting: 2b^2 = (2k)^2 = 4k^2. Dividing by 2: b^2 = 2k^2. This means b^2 is even, so b must be even. But if a and b are both even, they have 2 as a common factor, contradicting our assumption. Therefore, √2 cannot be rational.

Proposed Prompt Input (PSR; Stage 1: High-level Sketch)

Generate a high-level proof sketch for the theorem: The square root of 2 is irrational.

Proposed Prompt Expected Output (PSR; Stage 1: High-level Sketch)

Proof Sketch: 1) Assume √2 is rational. 2) Express √2 as a fraction in lowest terms. 3) Square both sides of the equation. 4) Show that this leads to a contradiction where both numerator and denominator must be even. 5) Conclude that the assumption is false, so √2 is irrational.

Proposed Prompt Input (PSR; Stage 2: Intermediate Formalization)

Expand and formalize the following proof sketch: 1) Assume √2 is rational. 2) Express √2 as a fraction in lowest terms. 3) Square both sides of the equation. 4) Show that this leads to a contradiction where both numerator and denominator must be even. 5) Conclude that the assumption is false, so √2 is irrational.

Proposed Prompt Expected Output (PSR; Stage 2: Intermediate Formalization)

1) Assumption: √2 is rational. 2) By definition of rational numbers, √2 = a/b, where a and b are integers with no common factors. 3) Square both sides: 2 = a^2/b^2 4) Multiply by b^2: 2b^2 = a^2 5) Since 2b^2 is even, a^2 must be even. 6) For a^2 to be even, a must be even. So a = 2k for some integer k. 7) Substitute: 2b^2 = (2k)^2 = 4k^2 8) Divide by 2: b^2 = 2k^2 9) This means b^2 is even, so b must be even. 10) But if both a and b are even, they have a common factor of 2, contradicting step 2. 11) Therefore, the assumption that √2 is rational must be false. 12) Conclusion: √2 is irrational.

Proposed Prompt Input (PSR; Stage 3: Formal Proof Generation)

Convert the following intermediate proof into a formal proof representation: [Insert the output from Stage 2]

Proposed Prompt Expected Output (PSR; Stage 3: Formal Proof Generation)

Theorem: √2 is irrational Proof: 14. ∴ √2 is irrational Q.E.D.

- Assume (for contradiction): ∃a,b ∈ ■, gcd(a,b) = 1, √2 = a/b
- (a/b)^2 = 2
- a^2 = 2b^2
- a^2 ≡ 0 (mod 2)
- a ≡ 0 (mod 2)
- ∃k ∈ ■, a = 2k
- (2k)^2 = 2b^2
- 4k^2 = 2b^2
- 2k^2 = b^2
- b^2 ≡ 0 (mod 2)
- b ≡ 0 (mod 2)
- 2 | a ∧ 2 | b
- Contradiction with gcd(a,b) = 1
- ∴ √2 is irrational

Proposed Prompt Input (PSR; Stage 4: Verification and Refinement)

Verify the correctness of the following formal proof and suggest any necessary modifications: [Insert the output from Stage 3]

Proposed Prompt Expected Output (PSR; Stage 4: Verification and Refinement)

Verification Result: The formal proof is correct and complete. No modifications are necessary. Each step logically follows from the previous ones, and the proof successfully demonstrates the irrationality of √2 through contradiction.

explanation

The PSR method breaks down the proof generation process into manageable stages, allowing for a more structured and refined approach. The high-level sketch provides a clear outline, which is then expanded and formalized. The formal proof generation stage converts this into a rigorous mathematical representation, which is finally verified for correctness. This multi-stage approach allows for better handling of complex proofs compared to direct generation methods.

Fallback Plan

If the proposed PSR method does not significantly outperform baseline methods, we can conduct a detailed analysis to understand why. This could involve examining each stage of the PSR process to identify bottlenecks or failure points. We might find that certain types of theorems benefit more from this approach than others, leading to a more nuanced understanding of when to apply PSR. Additionally, we could explore hybrid approaches that combine elements of PSR with other proof generation methods. Another direction could be to focus on improving the intermediate stages, such as developing more sophisticated models for proof refinement or enhancing the verification process. If the multi-stage approach proves challenging, we could pivot to investigating how different prompting strategies for large language models affect their ability to generate mathematical proofs, turning this into a study on effective prompting for mathematical reasoning.

Ranking Score: 6