Automated Theorem Proving with Neural Conjecturing - DMP Proposal

Charlie Meyer—abs6bd@virginia.edu

April 2025

Abstract

Automated Theorem Proving (ATP) remains a key challenge for language models, especially with proving formal mathematics in environments like Lean. In this work, I propose my DMP project, a new approach to ATP systems that conbines neural solvers with a neural conjecturer, using an adversarial PAIRED paradigm. My system will use supervised fine-tuning, online reinforcement learning, and regret-based theorem generation to build a generalizable ATP agent. This project aims to advance ATP towards superhuman mathematical reasoning, with broader implications for AI alignment and general intelligence.

1 Introduction

Mathematical reasoning has posed major challenges to language models (LMs) both for reasoning steps and identifying final answers [15]. However, the development of formal languages like Lean [7] provide a verifiable source of truth for a mathematical proof, unlike the unformalized question and response space for LM benchmarking. The inability to generalize basic mathematical proof techniques stems from a lack of formal Lean data (the most popular Lean4 library, Mathlib, contains 206k theorems and 103k definitions). Despite attempts to turn natural language mathematical problems into formalized Lean theorems through a process of autoformalization and use it as training data, modern automated theorem proving (ATP) systems have yet to overcome the lack of data and poor generalization.

Current product-based use cases for ATP include Lean Copilot [16], similar to other AI-powered assistants for programming and ideation. However, the implications of building superhuman mathematical competence extends beyond a copilot. With a mathematically super intelligent system, ATP systems can verify the outputs of AI-generated code, stopping hallucinations and creating provably safe software verification.

Outside product-based applications, superintelligent mathematical systems will quickly advance mathematics outside of a proof copilot. Because of the fallibility of human proofs (for example, it was long believed that Hilbert's 21st

problem was solved until a counterexample was found 81 years later), using formal languages like Lean enable us to have absolute certainty on the accuracy of a proof. After this, the implications of solving *unsolved* conjectures will radically transform the field of mathematics. Imagine a world in which a system can solve a millenium problem!

Finally, I believe that mathematical superintelligence is a sufficient (but not necessary) condition for artificial general intelligence (AGI), and I believe working on safe and aligned AGI is the most important problem to work on. Therefore, this sub-problem of AGI is incredibly important.

2 Related Work

While ATP systems have existed for a while, such as formalization manually through *interactive theorem provers*, early systems were primarily constructed on human heuristics and were unable to generalize. These early projects relied on formal languages like Lean [7] and Coq [13]. However, the advancements in LMs and reinforcement learning (RL) are indicators that generalizable ATP is tractable. The first explorations into ATP and autoformalization came through Wu et al. [18], whose goal was to extract useful formal training data from natural language mathematical data. Despite improvements in data and benchmarking systems through papers like ProofNet [1], the popularity in neural theorem proving increased with the proliferation of popular open-weight models. One of the first attempts at automated theorem proving was GPT-F [11], which both proved formal theorems and contributed to the Metamath library. GPT-F showed that supervised fine-tuning (SFT) on both informal and formal mathematical data coupled with an implementation of Monte-Carlo tree search (MCTS) over proof states can significantly improve a neural theorem prover.

After GPT-F, through improvements in LMs like GPT-4o [8] and reasoning models like DeepSeek-R1 [2], foundational model labs have found significant success in using pure language models for whole proof generation. DeepSeek-R1 successfully completed a single proof on PutnamBench with a pass@1, and newer models from OpenAI - GPT 4o and o4-mini-high [9], have completed one and two proofs on PutnamBench, respectively. Through these rapid advancements, many labs have attempted neural theorem proving using both traditional LMs and reasoning models. DeepSeek-Prover [20] achieved a state-of-the-art on MiniF2F by generating synthetic data and fine-tuning DeepSeek Math [14] 7B on the synthetic data. In addition to the advancements in LM-based neural theorem proving, DeepSeek-Prover-V1.5 [19] paired SFT of DeepSeek-Prover-V1 [20] with reinforcement learning from proof assistant feedback (RLPAF) with MCTS to achieve state of the art on both MiniF2f and ProofNet.

Reinforcement learning-based approaches then have become standard for modern neural theorem proving models. Most recently, ABEL [3] leveraged hyper-tree proof search (HTPS) [5] and online RL in a sample-efficient manner (the online RL utilized only 244 problems from MiniF2F-valid). A preview of Kimina-Prover [4] was just released on April 14th which achieved a SOTA of

10 problems solved on PutnamBench (pass@192), and impoved upon previous ATP systems by leveraging reasoning-driven exploration (instead of MCTS or best-first search) to imitate human reasoning patterns in mathematical proofs. Moreover, on April 25th, Kimina just open-sourced their Lean language server [12], an improvement over the Lean community's REPL [6], along with PR's being merged into the REPL to solve parallel tactic application on April 28th!

3 Plan

I will build an ATP system using both neural solvers and a neural conjecturer, leveraging an adversarial protaganist-angagonist PAIRED paradigm [10]. In this paradigm, a conjecturer will generate a proof to solve, and then the protagonist and antagonist play through this proof environment. By adding this constraint, the conjecturer will propose feasible theorems that are challenging yet tractable. This is to avoid the situation in which a conjecturer proposes incredibly difficult or unsolvable theorems, which would not help the solving agents learn. To do this, the adversary will learn to propose theorems that maximize regret (the difference in rewards between the protagonist and antagonist), so the theorems will get progressively hard as the proving agents learn. This is a novel approach that has not yet been attempted by any existing ATP system.

3.1 ATP System Outline

The system contains the core components:

- Conjecturer - the agent that proposes conjectures, parameterized by θ_c.
- Protagonist and Antagonist - the agents that solve the conjectures and provides a critic value on the proveability of a given proof state, parameterized by θ_p and θ_a, respectively.
- Replay Buffer - to store the (state, action, state) results, a centralized replay buffer will be used to train the agents.
- Solver Module - the main interface that will interact with Lean's kernel to provide a new proof state given a tactic applied to a proof state.
- Search Module - the module that contains a solver module which will perform proof tree search to guide the agents.

3.2 Training

3.2.1 SFT

I will begin the training process by performing supervised fine-tuning of a base model (likely Kiminia-Prover 7B) on a set of data from Mathlib. This will allow me to format the LM's responses to effectively extract the tactic to apply and the critic's evaluation of the current proof state. I will also perform a hyperparameter sweep to avoid catastrophic forgetting or overfitting.

3.2.2 Online RL

After fine-tuning, we will begin the online self-play training. This will occur wherein the conjecturer proposes theorems, and then the solver agents attempt to solve them via proof search. At each step in proof search, the agents will generate N tactics to apply, and then have their search guided by the neural critic. The conjecturer will be incentivized to propose difficult but feasible theorems through maximizing regret.

The replay buffer will store transitions of (state, action, next state) to facilitate training.

3.2.3 Checkpointing

The system will save model snapshots of the agents, and at a specific number of steps, we will evaluate the agents on a validation set from MiniF2F. Later into the training loop, we will evaluate against PutnamBench.

4 Evaluation

There are several popular benchmarks to evaluate automated theorem proving systems, most popular being PutnamBench [17] and MiniF2F [21]. As of the time of writing, the best models get 10 out of 657 (pass@192) on PutnamBench and above 80% on MiniF2F. For the purposes of this proposal, we will not leverage ProofNet as an evaluation metric, for it is primarily used for autoformalization benchmarking.

Although MiniF2F is a popular benchmark, top ATP systems have performed online RL on subsets of MiniF2F (ABEL performed online RL on a subset of MiniF2F-valid, and DeepSeek-Prover-V1.5 included MiniF2F-valid as a part of its SFT). Therefore, certain derivative models that use DeepSeekProver as a base model may should be skeptial of MiniF2F results. Therefore, the best evaluation metric of modern ATP systems is through PutnamBench.

To determine the success of my model, I will evaluate my model primarily through PutnamBench, including pass@1, 10, 512, and 4096 (popular pass@k metrics from other models).

5 Challenges

There are many non-trivial (and expensive) engineering challenges one must overcome to make a high-quality test of this ATP system:

- Computationally expensive hyperparmeter sweeps, along with testing different types of fine-tuning.
- Parallelized tactic application.
- Maximizing GPU utilization for both generation and training.

- Tuning online RL parameters, including sampling from the replay buffer and tuning the conjecturer.

6 Asks, Help

Although papers like ABEL showed that sample-efficient methods with *relatively* cheap training and evaluation costs can achieve SOTA results on benchmarks, the addition of neural conjecturing and a protagonist-antagonist paradigm will significantly increase the cost of training.

Because of this, I would like to explore various funding sources. Renaissance Philanthropy introduced a new "AI for Math" fund formal verification/ATP. The application cycle for this fund has closed, but funding by XTX Markets is a tractable way to receive a research grant. There exist other grant opportunities run by various companies and institutions (Cohere, OpenAI NextGenAI, Simons Foundation, DARPA) that I will apply for, as well. I would also like to explore various grants within the University, including through the Office of Undergraduate Research, the Jefferson Trust, Strategic Research Themes, and alternative sources like the E-Cup to fund a large training run. I would also request help from the UVA research community to request guidance in finding other sources of funding on Grounds that would be in support of this research.

Moreover, I will likely work with the HPC community on Grounds to see how many GPUs I can access from Rivanna.

References

[1] Zhangir Azerbayev et al. *ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics*. Issue: arXiv:2302.12433 _eprint: 2302.12433. Feb. 2023. DOI: 10.48550/arXiv.2302.12433. (Visited on 04/22/2025).

[2] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. Issue: arXiv:2501.12948 _eprint: 2501.12948. Jan. 2025. DOI: 10.48550/arXiv.2501.12948. (Visited on 04/22/2025).

[3] Fabian Gloeckle et al. "ABEL: Sample Efficient Online Reinforcement Learning for Neural Theorem Proving". In: *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*. Oct. 2024. (Visited on 04/22/2025).

[4] *Kimina-Prover-Preview/Kimina_Prover_Preview.Pdf at Master \cdot MoonshotAI/Kimina-Prover-Preview*. Publication Title: GitHub. URL: https://github.com/MoonshotAI/Kimina-Prover-Preview/blob/master/Kimina_Prover_Preview.pdf (visited on 04/22/2025).

[5] Guillaume Lample et al. *HyperTree Proof Search for Neural Theorem Proving*. Issue: arXiv:2205.11491 _eprint: 2205.11491. May 2022. DOI: 10.48550/arXiv.2205.11491. (Visited on 04/22/2025).

[6] *leanprover-community/repl*. original-date: 2023-03-30T23:12:19Z. Apr. 2025. URL: https://github.com/leanprover-community/repl (visited on 04/26/2025).

[7] Leonardo de Moura and Sebastian Ulrich. *The Lean 4 Theorem Prover and Programming Language \textbar SpringerLink*. URL: https://link.springer.com/chapter/10.1007/978-3-030-79876-5_37 (visited on 04/22/2025).

[8] OpenAI et al. *GPT-4o System Card*. arXiv:2410.21276 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2410.21276. URL: http://arxiv.org/abs/2410.21276 (visited on 04/26/2025).

[9] *OpenAI o3 and o4-mini System Card*. en-US. URL: https://openai.com/index/o3-o4-mini-system-card/ (visited on 04/26/2025).

[10] *PAIRED: A New Multi-agent Approach for Adversarial Environment Generation*. en. URL: https://research.google/blog/paired-a-new-multi-agent-approach-for-adversarial-environment-generation/ (visited on 04/26/2025).

[11] Stanislas Polu and Ilya Sutskever. *Generative Language Modeling for Automated Theorem Proving*. arXiv:2009.03393 [cs]. Sept. 2020. DOI: 10.48550/arXiv.2009.03393. URL: http://arxiv.org/abs/2009.03393 (visited on 04/26/2025).

[12] *project-numina/kimina-lean-server: Kimina Lean server*. URL: https://github.com/project-numina/kimina-lean-server (visited on 04/26/2025).

[13] *Rocq, formally Coq*. en. Mar. 2025. URL: https://rocq-prover.org (visited on 04/26/2025).

[14] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. arXiv:2402.03300 [cs]. Apr. 2024. DOI: 10.48550/arXiv.2402.03300. URL: http://arxiv.org/abs/2402.03300 (visited on 04/26/2025).

[15] Ali Shehper et al. *What Makes Math Problems Hard for Reinforcement Learning: A Case Study*. Issue: arXiv:2408.15332 _eprint: 2408.15332. Feb. 2025. DOI: 10.48550/arXiv.2408.15332. (Visited on 04/22/2025).

[16] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. *Lean Copilot: Large Language Models as Copilots for Theorem Proving in Lean*. arXiv:2404.12534 [cs]. Mar. 2025. DOI: 10.48550/arXiv.2404.12534. URL: http://arxiv.org/abs/2404.12534 (visited on 04/24/2025).

[17] George Tsoukalas et al. *PutnamBench: Evaluating Neural Theorem-Provers on the Putnam Mathematical Competition*. Issue: arXiv:2407.11214 _eprint: 2407.11214. Nov. 2024. DOI: 10.48550/arXiv.2407.11214. (Visited on 03/10/2025).

[18] Yuhuai Wu et al. *Autoformalization with Large Language Models*. Issue: arXiv:2205.12615 _eprint: 2205.12615. May 2022. DOI: 10.48550/arXiv.2205.12615. (Visited on 03/10/2025).

[19] Huajian Xin et al. *DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search*. Issue: arXiv:2408.08152 _eprint: 2408.08152. Aug. 2024. DOI: 10.48550/arXiv.2408.08152. (Visited on 04/22/2025).

[20] Huajian Xin et al. *DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data*. Issue: arXiv:2405.14333 _eprint: 2405.14333. May 2024. DOI: 10.48550/arXiv.2405.14333. (Visited on 04/22/2025).

[21] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. *MiniF2F: A Cross-System Benchmark for Formal Olympiad-level Mathematics*. Issue: arXiv:2109.00110 _eprint: 2109.00110. Feb. 2022. DOI: 10.48550/arXiv.2109.00110. (Visited on 03/10/2025).