AI4Math

Riccardo Valperga
Hi!
r.valperga@uva.nl

1. Introduction

The intersection of Artificial Intelligence (AI) and mathematics has emerged as a dynamic and rapidly expanding field, often referred to as AI for Mathematics (AI4Math). Within this domain, the application of LLMs has garnered considerable attention, initially focusing on intuitive or informal mathematics. However, there is a growing recognition of the potential of LLMs to contribute to the more rigorous realm of formal mathematics, where the emphasis is on the *machine-verifiable* correctness of proofs. As mathematical problems become increasingly complex, the need for advanced tools that can aid in verification and facilitate collaboration becomes critical, positioning formal languages as a potentially transformative technology for the future of mathematical research.

But why should we care? Despite being relatively young, the project has already lead to big breakthroughs. One notable example is the ambitious project to formalize the proof of Fermat's last theorem. While working on the formalization, an error was spotted in a section of an old proof that forms the foundations of crystalline cohomology. The author appeared to have forgotten a symbol between one line and another, invalidating the proof [1].

2. Lean

Developed at Microsoft Research starting in 2013, Lean [1] is an open-source proof assistant and programming language designed for formal, machine-checkable mathematics. It is supported and advanced by an active global community of mathematicians. A proof assistant is a software tool designed to help users write, check, and verify mathematical proofs through formal logic and computation. Lean provides the underlying language and logical framework. Built upon the principles of dependent type theory and supported by an extensive mathematical library known as Mathlib, Lean provides a robust platform for exploring the capabilities of LLMs in the context of formal mathematics.

3. Mathlib

Mathlib[2] is a comprehensive library of formal mathematical definitions, theorems, and proofs, spanning many areas of mathematics (algebra, analysis, topology, number theory, category theory, geometry, logic, etc.). It is a community-diven project that contains the formalization of a vast range of mathematical knowledge, from undergraduate curriculum to cutting-edge research. Its current size exceeds 1.5 million lines of code encompassing over 165k theorems and 85k definitions, and represents an invaluable resource for mathematicians and researchers seeking to ensure the rigor and correctness of their work. For context, figure 1 provides a simple example of a theorem formalized in Lean 4.

4. The Project

Among the various potential projects at the intersection of AI and formal mathematics, the development of an advanced semantic search engine for Mathlib presents arguably the safest yet most promising direction for this course. This claim is motivated by a significant and frequently expressed need within the Lean community[3] for more effective tools to navigate the vast Mathlib library. According to community feedback, existing search methods often prove inadequate, struggling to

[1] https://www.microsoft.com/en-us/research/project/lean/
[2] https://leanprover-community.github.io/mathlib-overview.html
[3] See the Zulip chat of the Lean community: https://leanprover.zulipchat.com/

```
import MIL.Common
import Mathlib.Data.Real.Basic

theorem mul_comm_assoc (a b c : R) : c * b * a = b * (a * c) := by
  rewrite [mul_comm c b]
  rewrite [mul_assoc]
  rewrite [mul_comm c a]
```

Figure 1. Example of Lean code. Proof that for any three real numbers a, b, c the product of $(a * b) * c = a * (b * c)$. It starts by stating the theorem, then, the `rewrite` is employed to apply the commutativity of multiplication (`mul_comm`) to a and b and associativity of multiplication (`mul_assoc`) to a and c, and finally commutativity again to obtain the theorem statement and finalize the proof.

capture the semantic nuances of mathematical queries and thus hindering efficient knowledge discovery. Therefore, building a superior search tool directly addresses a clear gap and offers a high-impact contribution.
Beyond the development of a semantic search engine, *if the students involved in this project show sufficient motivation and and some prior experience with foundation models and RL techniques*, we can also explore more advanced research directions. These could include investigating the application of LLMs for tactic suggestion within the Lean environment tackling the complex task of autoformalization – translating informal mathematics into Lean's formal language or even venturing into the realm of automated theorem proving by leveraging reinforcement learning techniques to guide the proof search process within Lean.

4.1. A Semantic Search Engine for Mathlib

The sheer scale and complexity of Mathlib present a significant challenge for users, both newcomers and experienced formalizers, when trying to locate specific theorems, definitions, or proofs relevant to their current task. Often, users must rely on familiarity with naming conventions, browsing through module structures, or resorting to keyword-based searches within the codebase, which can be inefficient and may not always yield the most semantically relevant results. This difficulty in navigating the library can hinder the process of mathematical formalization, potentially slowing down research and increasing the learning curve for individuals new to Lean. To address this critical need, we propose the development of an intelligent search engine specifically designed for the Lean theorem prover and its Mathlib library. The goal of this research project is to create a tool that allows users to efficiently search for mathematical knowledge within Mathlib using natural language queries or even by providing a snippet of a proof state. Such a tool would be invaluable to the Lean community, significantly enhancing productivity, facilitating knowledge discovery, and lowering the barrier to entry for new users.

This research will focus on leveraging state-of-the-art information retrieval techniques, particularly those involving LLMs [2], to build a semantic search engine capable of understanding the intent behind user queries and retrieving the most relevant mathematical entities from Mathlib. We aim to improve upon existing efforts in this direction, primarily [3] and [4] by exploring advanced embedding models and search strategies that can effectively capture the nuanced relationships between mathematical concepts as expressed in the Lean formal language. Furthermore, a key requirement for the proposed search engine is its ability to run locally on the user's machine, directly interacting with the version of Mathlib installed in their Lean environment. This local operation ensures seamless integration with the user's workflow and eliminates the need for external network dependencies.

4.2. Beyond the Search Engine: AI for Formal Mathematics

This project proposal primarily focuses on developing an advanced semantic search engine for Mathlib, with the possibility for exploring more advanced, *high-risk-high-reward* research directions, contingent on students' motivation and experience. The possibilities are many. Here I outline the most popular ones for which benchmarks already exist in the literature. For a complete survey on AI for formal mathematical reasoning see [5].

Autoformalization Autoformalization is the task of converting informal mathematical statements and proofs written in natural language into formal languages like Lean, is a fundamental prerequisite for effectively utilizing LLMs in formal mathematics. Several research initiatives have concentrated on creating datasets and benchmarks specifically designed to assess the autoformalization capabilities of LLMs within the Lean ecosystem. The FormL4 benchmark [6], for instance, is

specifically tailored for evaluating autoformalization in Lean 4, addressing the challenges arising from the language's ongoing evolution and its relatively limited presence in existing training datasets. In a related effort, an evaluation benchmark was introduced to test the performance of state-of-the-art LLMs, including GPT-3.5, GPT-4, and Gemini Pro, on their ability to autoformalize mathematical statements into Lean 4. The findings revealed that even these advanced models still face limitations when dealing with more complex areas of mathematics. TheoremLlama [7] represents another significant contribution, proposing an end-to-end framework aimed at training a general-purpose LLM to achieve expertise in Lean 4, which includes methods for generating datasets aligned with both natural and formal language.

Tactic suggestion Another key application of LLMs in formal mathematics involves their integration within the Lean environment to provide suggestions for the next steps in a proof, known as tactics. The goal of this integration is to make the process of developing formal proofs more accessible and efficient for users . Several tools have been developed to facilitate this, including Lean Copilot and LLMSTEP [8], which allow users to harness the power of LLMs for tactic suggestions directly within their Lean development environment. Lean Copilot [9] is a comprehensive framework that enables running LLM inference natively within Lean, offering functionalities not only for tactic suggestion but also for proof search and premise selection. LLMSTEP, on the other hand, functions as a Lean 4 tactic that sends the current proof state to a server hosting a language model, which then generates potential next tactics that are subsequently checked for validity by Lean.

Reinforcement Learning for Formal Mathematics Reinforcement learning [10] is increasingly being explored as a promising paradigm for training LLMs to tackle the complex task of formal theorem proving within the Lean environment. In this context, the verification outcomes provided by the proof assistant can serve as a robust reward signal, guiding the learning process of the LLM. ABEL [11] is a notable example of a scalable and efficient online RL framework specifically designed for theorem proving in Lean. It has demonstrated near state-of-the-art performance while requiring only a limited amount of training data, utilizing techniques such as hypertree proof search (HTPS) [12] and AlphaZero-style tree search [13]. LeanListener [14] is another framework that employs feedback directly from Lean within a reinforcement learning loop to optimize the generation of tactics. OpenAI has also contributed to this area by releasing lean-gym [15], a reinforcement learning environment built upon the Lean theorem prover. Furthermore, Proof Decomposer (ProD) represents an RL-based approach that encourages LLMs to adopt a strategy similar to human mathematicians by decomposing a given theorem into smaller, more manageable lemmas, proving these lemmas, and subsequently using them to prove the original theorem .

The integration of RL with Lean provides a powerful training environment for LLMs because the correctness of their actions, in this case, the application of tactics, can be directly and unambiguously verified by the Lean proof assistant. This direct verification yields a strong and well-defined reward signal for the learning process. This is a significant advantage over traditional language modeling approaches, where the notion of correctness is often more implicit or challenging to define precisely. Frameworks like ABEL showcase the potential of RL to achieve high levels of performance in automated theorem proving while requiring significantly less training data compared to purely supervised learning methods. This suggests that RL may offer a more data-efficient pathway to training these complex models. The interactive nature of reinforcement learning, where an agent learns through a process of trial and error and receives immediate feedback from the environment, can be particularly effective for mastering intricate tasks like theorem proving, potentially overcoming the data scarcity challenges that are prevalent in this domain.

5. Conclusions

In conclusion, this research addresses a clearly identified need within the formal mathematics community. current approaches are often considered insufficient by practitioners, particularly those working with advanced proof assistants like lean, indicating a significant gap and opportunity for impactful innovation. the successful development of the proposed methods promises substantial contributions, advancing a field recognized by leading researchers.

Furthermore, the intersection of formal mathematics and artificial intelligence is rapidly emerging as a frontier with transformative potential. methodologies such as reinforcement learning are poised to be particularly relevant, especially as we strive to develop systems capable of tackling mathematical complexities beyond current human limits. achieving this goal necessitates moving beyond paradigms reliant solely on human-generated data, making the exploration proposed herein both timely and critical for future progress in automated reasoning and mathematical discovery.

To conclude, let me add that many notable researchers in the field of Mathematics, such as Terence Tao[4] [5] have already expressed themselves very positively about this research direction. Big tech companies such as Google, Meta, AWS and OpenAI have all started working on mathematical reasoning. Let's jump on that train!

References

[1] Matthew Boyle. Exclusive – donald trump: Paul ryan "doesn't know what he's talking about" on anchor babies. https://archive.ph/IbtvL, August 2015. Archived version retrieved from archive.ph. Original publication date: 2015-08-19. 1

[2] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023. 2

[3] Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, and Bin Dong. A semantic search engine for mathlib4. *arXiv preprint arXiv:2403.13310*, 2024. 2

[4] Yicheng Tao, Haotian Liu, Shanwen Wang, and Hongteng Xu. Assisting mathematical formalization with a learning-based premise retriever. *arXiv preprint arXiv:2501.13959*, 2025. 2

[5] Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. Formal mathematical reasoning: A new frontier in ai. *arXiv preprint arXiv:2412.16075*, 2024. 2

[6] Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, et al. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024. 2

[7] Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts. *arXiv preprint arXiv:2407.03203*, 2024. 3

[8] Sean Welleck and Rahul Saha. Llmstep: Llm proofstep suggestions in lean. *arXiv preprint arXiv:2310.18457*, 2023. 3

[9] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024. 3

[10] Kevin Murphy. Reinforcement learning: An overview. *arXiv preprint arXiv:2412.05265*, 2024. 3

[11] Fabian Gloeckle, Jannis Limperg, Gabriel Synnaeve, and Amaury Hayat. Abel: Sample efficient online reinforcement learning for neural theorem proving. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. 3

[12] Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in neural information processing systems*, 35:26337–26349, 2022. 3

[13] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 3

[14] Sara Rajaee, Kumar Pratik, Gabriele Cesa, and Arash Behboodi. Local look-ahead guidance via verifier-in-the-loop for automated theorem proving. *arXiv preprint arXiv:2503.09730*, 2025. 3

[15] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022. 3

[4] https://www.youtube.com/watch?v=AayZuuDDKP0
[5] https://www.youtube.com/watch?v=e049IoFBnLA