A Comprehensive Review of Large Language Models in Automated Theorem Proving: From Autonomous Systems to Neuro-Symbolic Copilots

Mayank Pareek
mpareek@depaul.edu
Department of Mathematical Sciences
Depaul University
CHICAGO, IL, USA

Tulsi Patel
tpatel91@depaul.edu
Department of Mathematical Sciences
Depaul University
CHICAGO, IL, USA

Abstract

The confluence of large language models (LLMs) and formal systems marks a transformative era in automated theorem proving (ATP). Historically, ATP has been a formidable challenge for artificial intelligence, primarily confined to highly structured, symbolic domains. The recent advent of LLMs, with their unparalleled ability to process and generate complex linguistic patterns, has introduced a new paradigm, enabling the translation of human-centric mathematical concepts into machine-verifiable proofs. This report provides a detailed review of the current landscape, analyzing two primary research directions: the quest for fully autonomous provers and the development of neuro-symbolic AI copilots. The analysis reveals a strategic shift away from monolithic, autonomous systems, which often struggle with novel or complex problems, toward collaborative frameworks that augment human expertise. Key to this evolution are innovative methodologies such as retrieval-augmented generation (RAG) to address the critical bottleneck of premise selection, and modular, subgoal-based approaches that scaffold the LLM's reasoning process. The report synthesizes performance metrics from state-of-the-art systems on major benchmarks like MiniF2F and CoqGym, demonstrating that combining the probabilistic intuition of LLMs with the absolute rigor of proof assistants is the most viable path to advancing the field. It concludes by highlighting the remaining challenges and outlining a future trajectory focused on creating a positive feedback loop between AI and human mathematicians, ultimately accelerating the formalization of global mathematical knowledge.

1. Introduction: The New Frontier of AI-Assisted Formal Reasoning

The field of automated theorem proving represents a long-standing grand challenge in artificial intelligence, with roots tracing back to the foundational work of computer science. The core objective is to mechanize the process of logical deduction and mathematical proof, a task long considered the pinnacle of human cognition. Traditional approaches to ATP have been built on symbolic logic and rule-based systems, which, while powerful in their specific domains, often lack the flexibility and broad knowledge required to handle the ambiguity and complexity of human-authored mathematics. For decades, the gap between informal mathematical discourse and machine-verifiable formal systems appeared insurmountable, creating what has been described as a "translation paradox".

This landscape has been fundamentally altered by the emergence of large language models. Trained on vast corpora of text and code, LLMs exhibit a surprising capacity for understanding and generating content across a wide range of domains, including formal languages and mathematics.[2] This capability has opened a new frontier, allowing researchers to explore the potential of a "neuro-symbolic" approach that combines the probabilistic, pattern-matching strengths of LLMs with the absolute, verifiable rigor of formal systems.[3] The current research is centered on two primary paradigms: the use of LLMs for *autoformalization*, which is the translation of natural language mathematics into a formal, machine-readable language, and for *direct proof generation*, where LLMs are used to generate the proof steps themselves.

This report provides a comprehensive analysis of these developments, arguing that the most significant and sustainable progress is being achieved not by a single breakthrough, but through the synergistic integration of LLM capabilities with traditional computational logic. The evidence suggests that the future of this field lies in collaborative frameworks that leverage LLMs as powerful assistants, capable of intuiting solutions and automating tedious tasks, while relying on formal systems for infallible verification.

2. Foundational Concepts and Context

A deep understanding of the current research landscape requires a clear grasp of the core problems and the foundational tools used to address them. The primary challenge at the intersection of LLMs and mathematics is the inherent unreliability of LLM outputs. While capable of generating coherent and seemingly correct text, LLMs are known to "hallucinate" or produce factual errors, a fatal flaw in a domain where absolute correctness is non-negotiable.[4] A single incorrect step invalidates an entire proof.

2.1. The Role of Proof Assistants

This fundamental problem is resolved by the use of *proof assistants*, also known as *interactive theorem provers* (ITPs). These are software frameworks designed to help users construct mathematical proofs in a formal language that can be rigorously checked by a computer. Prominent examples include Lean, Isabelle/HOL, and Coq.[7] In these systems, a proof is written as a sequence of formal instructions, or "tactics," that transform a starting theorem state into a simpler, provable end state, often a tautology.[4] The proof assistant meticulously verifies each step, ensuring that every logical inference is sound and that no errors or hallucinations are introduced.[3] This synergy is paramount: the LLM can propose a proof, but the proof assistant provides the definitive, mechanically checked certification of its correctness.

A Lean proof can be analogized to a Python program, where the theorem is the task and each tactic is a command that modifies the program's state until a desired end state is reached.[4]

For example, consider the simple theorem: a * b * c = b * (a * c), where a, b, c are real numbers. A proof can be constructed with a sequence of tactics [4]:

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

This proof consists of two tactics:

- rw [mul_comm a b]: This tactic applies the commutative property of multiplication (mul_comm) to a and b, transforming the initial state a * b * c to b * a * c.
- rw [mul_assoc]: This tactic then applies the associative property (mul_assoc) to transform b * a * c into b * (a * c), which is the goal, thus finishing the proof.

2.2. Autoformalization: The Bridge Between Intuition and Precision

One of the most promising early applications of LLMs in this domain is *autoformalization*, defined as the process of automatically translating from natural language mathematics into formal specifications and proofs.[2] A successful autoformalization system would enable the absorption and verification of the vast body of human mathematical knowledge, revolutionizing fields from formal verification to program synthesis.[1]

Early foundational work demonstrated that LLMs possess a surprising ability to perform this translation. By using *in-context learning*, or few-shot prompting, researchers were able to show that models like PaLM and Codex could formalize mathematical statements without extensive fine-tuning.[2] One study made the unexpected observation that LLMs could correctly translate 25.3% of mathematical competition problems into formal specifications in Isabelle/HOL.[2] This was particularly notable given that the amount of formal mathematics data in their training corpora was extremely limited, accounting for less than 0.18% of the data for a model like Codex.[2]

Despite these successes, autoformalization remains one of the most challenging problems in AI. The difficulty lies in what has been described as the "translation paradox".[1] Human mathematical language is laden with ambiguities, implicit assumptions, and contextual dependencies. When a mathematician writes, "Let G be a group," they assume a shared understanding of a vast web of associated concepts.[1] By contrast, a formal system requires every single detail to be explicitly specified and every inference rule to be stated.[1] As a result, natural language mathematical text contains both too much ambiguous information and too little explicit detail to be directly translatable.[1] This fundamental gap in representation is a major reason why LLMs, which are powerful pattern matchers, cannot simply formalize entire papers without a robust, iterative feedback loop with a formal system. This limitation underscores the need for the structured and collaborative approaches that now define the field.

3. The Two Paradigms: Autonomous Provers vs. AI Copilots

The research in LLM-based theorem proving has diverged into two principal paradigms, each with a distinct philosophy and set of challenges.

3.1. The Autonomous Approach: The Promise and the Pitfalls

The first approach pursues the long-term goal of an autonomous AI mathematician: a system that can prove theorems without any human intervention.[3] These systems are typically evaluated in a "gym-like environment" where the model interacts with a proof assistant on a backend server, with success measured by the number of theorems proven in a fully automated mode.[3] While desirable in the long run, this paradigm has encountered significant limitations. Current autonomous provers struggle with novel and complex theorems, particularly those from a domain not well-represented in their training data.[3] The lack of flexibility and human intuition makes it difficult for these systems to navigate complex proof spaces.

3.2. The Copilot Paradigm: A More Viable Path

In response to the limitations of the autonomous approach, a new, more pragmatic paradigm has emerged: the AI copilot.[3] This research direction is grounded in the observation that for complex problems, human insight and guidance are still critical. Rather than replacing the human mathematician, the goal is to augment their abilities by automating the more straightforward and tedious parts of the proof process.[3] This represents a strategic shift from a pure AI competition to a human-AI collaboration, where the metric of success is not just the number of theorems proven but the reduction in human effort.[3]

A prime example of this philosophy in action is **Lean Copilot**, a neuro-symbolic framework that integrates LLM inference directly into the Lean proof assistant environment.[3] It addresses the core technical challenge of running a neural network within a symbolic system by leveraging Lean's C++ Foreign Function Interface (FFI) to execute LLM inference natively.[3] This architecture avoids the overhead of inter-process communication and provides a suite of tools that assist the user [8]:

- suggest_tactics: proposes the next proof step (tactic).[8]
- search_proofs: attempts to construct a complete, verified proof by combining single tactic generation with a proof search algorithm.[8]
- select_premises: identifies relevant lemmas from the library.[8]

The efficacy of this collaborative approach is compellingly demonstrated by a comparison of Lean Copilot's performance against a traditional rule-based system (aesop) and a basic LLM-based tool (suggest_tactics). The data, presented in Table 1, shows that the combination of LLM-generated tactics with a proof search algorithm significantly outperforms both baselines.

Table 1: Copilot vs. Autonomous Performance on Mathematics in Lean [3]

Method	**Avg. # Human-Entered Tactics (↓)**	**% Proof Steps Automated (↑)**
aesop (Traditional)	3.86	40.1%
Lean Copilot (suggest_tactics)	3.10	58.3%
Lean Copilot (search_proofs)	2.08	74.2%

The data confirms that the copilot paradigm is not merely a theoretical alternative but a practical and highly effective one. The search_proofs tool reduced the number of human-entered tactics by 85% compared to aesop and automated 74.2% of proof steps, a 27% increase over the single-tactic suggest_tactics tool alone.[3] This significant improvement highlights the value of layering a search algorithm on top of the LLM's generative capabilities, providing a structured approach to solving the multi-step problem.

4. State-of-the-Art Methodologies and Performance Benchmarks

The field is currently characterized by a number of innovative methodologies that are pushing the boundaries of what is possible in LLM-assisted theorem proving.

4.1. The LeanDojo Framework and Retrieval-Augmented Provers

A significant barrier to progress in this domain has been the lack of open-source resources, with many existing LLM-based provers relying on "private code, data, and large compute requirements".[12] This has hindered reproducibility and broader research. The introduction of

LeanDojo was a pivotal moment, as it provided an open-source "playground" with toolkits, data, models, and benchmarks, democratizing access to state-of-the-art methods.[12] LeanDojo extracts fine-grained data from Lean, including proof trees and the premises used in each proof, to provide valuable data for premise selection, a key bottleneck in theorem proving.[12]

Built on this foundation is **ReProver (Retrieval-Augmented Prover)**, an LLM-based prover that explicitly addresses this bottleneck.[12] Proving a theorem often requires applying lemmas and definitions from a vast external library, and existing LLM-based provers often fail to identify the correct premises.[12] ReProver solves this by augmenting the LLM with a retrieval mechanism that first selects a handful of potentially useful premises from the library, which are then fed to the LLM along with the current proof state.[12] The system's retriever is based on Dense Passage Retriever (DPR) and is trained to find the most relevant premises by using a contrastive loss.[15] This approach is both effective and computationally efficient, requiring only one GPU week of training, thereby lowering the barrier to entry for research.[12] The system was shown to outperform GPT-4 in tactic generation and even found proofs for theorems that were missing proofs in the original library.[13]

4.2. Subgoal-Based and Modular Proving

As LLMs have been applied to more complex mathematical problems, it has become apparent that a more structured, granular approach is needed to manage their multi-step reasoning. Two cutting-edge systems exemplify this trend. **SubgoalXL** is a framework that synergizes subgoal-based proofs with expert learning to enhance LLMs' capabilities within the Isabelle environment.[16] By breaking down a complex proof into a series of smaller, manageable subgoals, the system provides a scaffold that guides the LLM through the logical steps, mimicking human problem-solving.[9] A similar, modular approach is taken by

LEGO-Prover, which constructs proofs in a "block-by-block manner" and employs a "growing skill library" of verified lemmas.[17] This allows the system to build on its own successes and learn reusable proof patterns, advancing its capabilities over time.[17]

This focus on structuring the proof process is a direct response to the inherent limitations of LLMs with complex, multi-step reasoning.[16] By providing a logical framework, these systems mitigate the risk of the model losing its way in a long proof chain.

4.3. The PALM Framework

The **PALM** framework provides another compelling example of the power of a structured, neuro-symbolic approach. It successfully integrates large, general-purpose LLMs with the Coq proof assistant to automate the theorem-proving process.[7] The performance of PALM on the large CoqGym dataset is a powerful validation of this methodology.

The results, synthesized in Table 2, provide a clear competitive landscape, highlighting the significant gap between standalone LLMs and those integrated into a robust framework.

Table 2: State-of-the-Art Performance on Key Benchmarks

System	**Methodology**	**Benchmark Dataset**	**Proof Rate / Success Rate**
PALM (GPT-4o)	LLM + Proof Automation Framework	CoqGym (10,842 theorems)	42.5% (4,614 theorems) [7]
LEGO-Prover	LLM + Modular "Skill Library"	MiniF2F-Valid	57.0% [17]

SubgoalXL	LLM + Subgoal-based Expert Learning	MiniF2F-Test	56.1% [16]
Seed-Prover	Lemma-style, iterative refinement	MiniF2F	Saturated the benchmark [18]
ReProver	Retrieval-Augmented LLM	LeanDojo Benchmark (98,734 theorems)	Outperforms GPT-4 [13]
Standalone LLMs (GPT-3.5, Llama)	Direct generation (no framework)	CoqGym (10,842 theorems)	3.7% - 3.6% [7]
Previous SOTA (FMSCL)	Expert Iteration on autoformalized data	MiniF2F	35.2% [2]

The data confirms that simply relying on a standalone LLM for proof generation is highly ineffective, with GPT-3.5 and Llama-3-70b-Instruct achieving a success rate of only 3.7% and 3.6%, respectively, on the CoqGym dataset.[7] By contrast, PALM with GPT-4o achieved a 42.5% success rate, proving 4,614 theorems and outperforming other state-of-the-art methods like Passport (14.4%) and DSP (22.9%).[7] Furthermore, the table demonstrates that subgoal-based and modular methods are achieving new state-of-the-art results on the MiniF2F benchmark, with LEGO-Prover and SubgoalXL reaching success rates above 56%.[16] The recently introduced Seed-Prover has "saturated" the MiniF2F benchmark, indicating a significant leap in performance.[18] These results collectively affirm that the most successful approaches are those that channel the generative power of large models through a robust, domain-specific framework.

5. Addressing Fundamental Challenges

Despite the rapid progress, the field is still grappling with several fundamental challenges that must be overcome to achieve more robust and generalizable AI reasoning.

5.1. The Data Scarcity Bottleneck

The primary challenge for LLMs in this domain is the scarcity of high-quality, human-written formal mathematics data on the web.[2] This is a significant barrier to training models that can reliably perform autoformalization and proof generation across diverse mathematical fields. For example, the Lean community's extensive Mathlib4 library, while a major resource, contains approximately 112,000 theorems, which still falls short of the scale required for optimal LLM training.[4]

5.2. Data Augmentation and Backtranslation

To overcome the data bottleneck, researchers are pioneering new data-centric solutions. A notable method is *backtranslation*, a technique borrowed from neural machine translation.[19] The process involves using a powerful LLM to translate formal proofs (the target language) into informal natural language (the source language).[19] This reverse translation creates a large, synthetic paired dataset that can then be used to fine-tune a "student" model on the more difficult formalization task.[19] The core idea is that LLMs are more proficient at informalizing a formal text than at formalizing an informal one.[21] This asymmetric capability provides a brilliant, indirect solution to the data scarcity problem, enabling the generation of high-quality, large-scale training corpora without extensive manual effort.[19]

5.3. Premise Selection and Retrieval

The critical bottleneck of premise selection is another major challenge. An analysis of the **PALM** framework's failures revealed that a key reason for failure (58% of cases) was the omission of necessary premises during the retrieval process.[7] For instance, a proof could fail simply because a critical premise was not retrieved, which prevented the LLM from using it.[7] Even when premises were successfully retrieved, the LLM failed to use them in 14% of cases.[7]

The prevalence of these failures provides strong empirical evidence that retrieval-augmented generation (RAG) is a necessary component for robust, complex theorem proving. The **ReProver** system, with its dedicated retrieval mechanism, is a direct response to this need.[12] By intelligently selecting a small set of potentially relevant premises and feeding them to the LLM, the system provides the model with the necessary "building blocks" to construct a valid proof. The fact that failures still occur when premises are provided but unused (as noted in the PALM analysis) highlights a more subtle challenge: the need for the LLM to not only identify but also strategically utilize the retrieved information.

6. Future Directions and Advanced Concepts

The current research points to a clear path forward, but also raises important open questions. The ongoing transition from autonomous provers to collaborative copilots is a testament to the value of combining human intuition with machine rigor. This hybrid, neuro-symbolic approach offers the most viable path to solving the grand challenge of mathematical reasoning.

Future research should focus on several key areas. The development of more sophisticated retrieval mechanisms is paramount. While current RAG systems are effective, the failure analysis of the PALM framework shows a need for systems that are more adept at providing context and encouraging the LLM to use the provided premises effectively.[7] The field should also explore the potential of multi-modal reasoning, perhaps by training models to interpret and utilize graphical or visual representations of mathematical concepts, which are integral to human understanding.[22]

6.1. Deep and Broad Reasoning

An emerging class of models is pushing the boundaries of multi-step reasoning by thinking "deeply and broadly".[18] The **Seed-Prover** model, for example, is a lemma-style reasoning model that iteratively refines its proof based on feedback from the Lean environment, previously proved lemmas, and self-summarization.[18] This iterative refinement process allows the model to correct its mistakes and build on its successes. To tackle complex problems, Seed-Prover uses three test-time inference strategies that enable both "deep and broad" exploration of the proof space.[18]

This structured, strategic approach has yielded impressive results:

- **IMO Problems:** Seed-Prover proved 5 out of 6 problems in the IMO 2025 competition.[18]
- **MiniF2F:** It "saturates" the MiniF2F benchmark, a key metric of success in the field.[18]
- **PutnamBench:** It achieved a proof rate of over 50% on PutnamBench, outperforming the previous state-of-the-art by a large margin.[18]

6.2. Integrating Informal Thoughts

Another promising avenue is the integration of informal thought processes into the formal proof generation. The **Lean-STaR** framework, for instance, trains language models to interleave informal thoughts with formal verification.[24] The underlying thought process, which is often not present in formal code, is captured and used to guide the model's reasoning.[24] This provides a layer of informal, human-like intuition that can guide the model through complex logical chains. The framework generates a "thought-augmented dataset" by synthesizing examples through expert iteration, teaching the model to think before it acts.[24]

6.3. Specialized Reasoning Engines

The research is also moving toward specialized AI engines for specific mathematical domains. To address the lack of geometry support in Lean, the **Seed-Geometry** engine was developed, which uses a dedicated domain-specific language and a C++ backend for massive performance increases (roughly 100x faster than a Python implementation).[18] This demonstrates that hybrid systems that combine general LLMs with highly specialized, purpose-built engines for difficult domains are a viable path forward.

7. Conclusion: Synthesis of Progress and Long-Term Vision

The journey of LLMs in automated theorem proving has been one of rapid evolution and strategic adaptation. The initial hypothesis that simply scaling language models would yield powerful provers proved insufficient, as standalone LLMs demonstrated low success rates and an inability to handle the logical rigor of formal systems. The field has since converged on a more nuanced, hybrid approach that treats LLMs not as monolithic, autonomous agents, but as powerful neuro-symbolic copilots.

The most significant progress has been driven by the synergistic combination of several methodologies: LLMs for generating plausible proof steps, formal proof assistants for providing infallible verification, retrieval mechanisms for solving the data-retrieval bottleneck, and structured approaches for managing the complexity of multi-step reasoning. The quantitative evidence from major benchmarks, such as the impressive proof rates of LEGO-Prover and SubgoalXL on MiniF2F and the substantial performance gains of the PALM framework, provides compelling validation of this trajectory. The emergence of systems like Seed-Prover and Lean-STaR signifies the next step, where models learn to reason more deeply, iteratively, and broadly.

The growth of this domain is a direct consequence of a fundamental shift in perspective: from attempting to replace the human mind's intuitive leap with brute-force computation, to building tools that augment and extend human capabilities. The long-term vision of an AI system that can absorb, verify, and expand the entirety of human mathematical knowledge is no longer a distant dream. With the continued development of open-source frameworks and a focus on solving the remaining challenges, a tangible path now exists to achieve this transformative goal, with profound implications for science, engineering, and artificial intelligence itself.

Works cited

1. Autoformalization: Bridging Human Mathematical Intuition and Machine Precision - Medium, accessed on September 22, 2025, https://medium.com/intuitionmachine/autoformalization-bridging-human-mathematical-intuition-and-machine-precision-cc00f9ff6c82
2. Autoformalization with Large Language Models - NIPS, accessed on September 22, 2025, https://papers.nips.cc/paper_files/paper/2022/file/d0c6bc641a56bebee9d985b937307367-Paper-Conference.pdf
3. Large Language Models as Copilots for Theorem Proving in Lean - arXiv, accessed on September 22, 2025, https://arxiv.org/html/2404.12534v2
4. Generating Millions Of Lean Theorems With Proofs By Exploring State Transition Graphs, accessed on September 22, 2025, https://arxiv.org/html/2503.04772v1
5. Benchmarking Automated Theorem Proving with Large Language Models - ACL Anthology, accessed on September 22, 2025, https://aclanthology.org/2024.nlp4science-1.18.pdf
6. Benchmarking Automated Theorem Proving with Large Language ..., accessed on September 22, 2025, https://aclanthology.org/2024.nlp4science-1.18/
7. Proof Automation with Large Language Models - arXiv, accessed on September 22, 2025, https://arxiv.org/pdf/2409.14274
8. [Literature Review] Lean Copilot: Large Language Models as Copilots for Theorem Proving in Lean - Moonlight, accessed on September 22, 2025, https://www.themoonlight.io/en/review/lean-copilot-large-language-models-as-copilots-for-theorem-proving-in-lean
9. SubgoalXL: Subgoal-based Expert Learning for Theorem Proving - Powerdrill AI, accessed on September 22, 2025, https://powerdrill.ai/discover/discover-SubgoalXL-Subgoal-based-Expert-cm05rsrh41f9d014k7seh95mz
10. Autoformalization with Large Language Models - OpenReview, accessed on September 22, 2025, https://openreview.net/forum?id=IUikebJ1Bf0
11. Towards Large Language Models as Copilots for Theorem Proving in Lean - Math-AI,

accessed on September 22, 2025, https://mathai2023.github.io/papers/4.pdf
12. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models, accessed on September 22, 2025, https://papers.neurips.cc/paper_files/paper/2023/file/4441469427094f8873d0fecb0c4e1cee-Paper-Datasets_and_Benchmarks.pdf
13. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models - OpenReview, accessed on September 22, 2025, https://openreview.net/forum?id=g7OX2sOJtn¬eId=EJxdCMebal
14. (PDF) LeanDojo: Theorem Proving with Retrieval-Augmented Language Models, accessed on September 22, 2025, https://www.researchgate.net/publication/371909561_LeanDojo_Theorem_Proving_with_Retrieval-Augmented_Language_Models
15. NeurIPS Poster LeanDojo: Theorem Proving with Retrieval-Augmented Language Models, accessed on September 22, 2025, https://neurips.cc/virtual/2023/poster/73510
16. SubgoalXL: Subgoal-based Expert Learning for Theorem Proving | OpenReview, accessed on September 22, 2025, https://openreview.net/forum?id=mb2rHLcKN5
17. Code for the paper LEGO-Prover: Neural Theorem Proving with Growing Libraries - GitHub, accessed on September 22, 2025, https://github.com/wiio12/LEGO-Prover
18. Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving - arXiv, accessed on September 22, 2025, https://www.arxiv.org/pdf/2507.23726v1
19. Autoformalization with Backtranslation: Training an Automated Mathematician - Stanford University, accessed on September 22, 2025, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1244/final-projects/JakobNordhagen.pdf
20. Autoformalization: Bridging Informal and Formal Math - Emergent Mind, accessed on September 22, 2025, https://www.emergentmind.com/topics/autoformalization
21. Exploration of neural machine translation in autoformalization of mathematics in Mizar, accessed on September 22, 2025, https://www.semanticscholar.org/paper/Exploration-of-neural-machine-translation-in-of-in-Wang-Brown/a3afb3795f7054fc5b334d2d6feb92f0999942c7
22. Could you give a an overview of automated theorem proving and some prediction on the future of it with machine learning techniques such as LLMs? - Quora, accessed on September 22, 2025, https://www.quora.com/Could-you-give-a-an-overview-of-automated-theorem-proving-and-some-prediction-on-the-future-of-it-with-machine-learning-techniques-such-as-LLMs
23. [2507.23726] Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving - arXiv, accessed on September 22, 2025, https://arxiv.org/abs/2507.23726
24. Lean-STaR: Learning to Interleave Thinking and Proving, accessed on September 22, 2025, https://arxiv.org/pdf/2407.10040