# Position: Formal Mathematical Reasoning—A New Frontier in AI

Kaiyu Yang [1] Gabriel Poesia [2] Jingxuan He [3] Wenda Li [4] Kristin Lauter [1] Swarat Chaudhuri [5] Dawn Song [3]

## Abstract

AI for Mathematics (AI4Math) is intellectually intriguing and is crucial for AI-driven system design and verification. Extensive efforts on AI4Math have mirrored techniques in NLP, in particular, training large language models on carefully curated math datasets in text form. As a complementary yet less explored avenue, *formal mathematical reasoning* is grounded in formal systems such as proof assistants, which can verify the correctness of reasoning and provide automatic feedback. This position paper advocates formal mathematical reasoning as an indispensable component in future AI for math, formal verification, and verifiable generation. We summarize existing progress, discuss open challenges, and envision critical milestones to measure future success.

## 1. Introduction

Mathematical reasoning has been important for AI from its early days (Newell & Simon, 1956) to the era of modern large language models (LLMs). It serves as a proxy for reasoning and planning tasks and plays a fundamental role in quantitative disciplines. AI4Math has the potential to revolutionize AI for science, engineering, and beyond.

Substantial research in AI4Math has focused on math LLMs. A common approach is to continue pretraining LLMs on mathematical data from the Web and finetune on curated math problems with detailed solutions. We call this the "informal" approach to distinguish it from the formal approach that will be introduced later. Math LLMs have a simple recipe, but the secret sauce is data curation. Carefully curated training data plus inference-time techniques, such as chain-of-thought (Wei et al., 2022), have led to remarkable success on benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). However, the success has been mostly limited to high school math, raising a key question: *How far can we go by scaling up the informal approach? Will it solve more challenging competition problems or even mathematical research problems?*

The informal approach faces challenges in dealing with advanced mathematics. First, high-quality training data is inherently scarce in advanced mathematics. Second, solutions to advanced problems are not limited to numbers but may include chains of intricate reasoning steps, such as proofs. LLMs are notorious for hallucinating seemingly valid reasoning steps, making it challenging to verify the correctness of model output or collect useful learning feedback. These challenges cannot be resolved by merely scaling up training. Researchers are exploring alternatives, such as scaling up inference (OpenAI, 2024), learning to reason via reinforcement learning (Guo et al., 2025), and neural verifiers (Cobbe et al., 2021). While these techniques have shown promise, we argue that neural networks alone are insufficient (Sec. 6). **A critical yet underexplored piece of the puzzle is formal mathematical reasoning. Combining formal reasoning with LLMs can significantly advance AI4Math, unlocking its applications in AI-driven formal verification and verifiable generation.**

We consider formal mathematical reasoning broadly as *mathematical reasoning grounded in formal systems*, including but not limited to higher-order logic (Nipkow et al., 2002), dependent type theory (Barras et al., 1997), and computer programs annotated with formal specifications (Leino, 2010). Formal systems provide environments that can verify the model's reasoning and provide automatic feedback. The feedback can mitigate data scarcity; also, such systems enable rigorous test-time checks that resist hallucination. In contrast, *informal mathematics* refers to math commonly found in textbooks and research papers. It interleaves natural language with symbols (e.g., LaTeX), but these symbols do not have a self-contained semantics, instead relying on informal text to convey significant parts of their meaning.

AlphaProof (Google DeepMind, 2024) and AlphaGeometry (Trinh et al., 2024) are two successful examples of this idea, leading to unprecedented performance in the International Mathematical Olympiad (IMO).[1] They build on a broad literature on the synergistic use of formal meth-

[1]Meta FAIR [2]Stanford University [3]UC Berkeley [4]University of Edinburgh [5]UT Austin. Correspondence to: Kaiyu Yang <kaiyuy@meta.com>, Dawn Song <dawnsong@cs.berkeley.edu>.

*Proceedings of the 42nd International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[1]Public information on AlphaProof's inner workings is limited.

ods and machine learning in mathematical tasks (Kaliszyk et al., 2018; Gauthier et al., 2021), including neural theorem proving, i.e., generating formal proofs given formal theorem statements, and autoformalization, i.e., automatically translating informal mathematics into formal mathematics. The advent of LLMs has significantly accelerated research in this area. For example, autoformalization was long hampered by the lack of aligned informal-formal data for finetuning. LLMs can mitigate this problem by either synthesizing the data (Jiang et al., 2024) or performing autoformalization without finetuning (Wu et al., 2022). LLMs are also powerful tools for theorem proving; in particular, recent approaches have used them to predict proof steps and fix buggy proofs (Thakur et al., 2024; First et al., 2023).

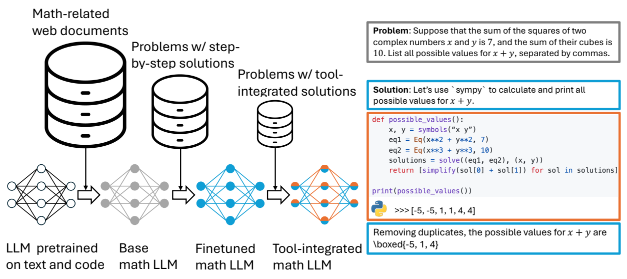
The research infrastructure around LLMs and formal reasoning is rapidly maturing. Lean (de Moura et al., 2015)—a language for writing formal proofs—has gained popularity among mathematicians. Multiple frameworks can support the interaction between LLMs and Lean (Yang et al., 2023; Aniva et al., 2024; Thakur et al., 2024). LLMs can assist humans in writing formal proofs (Song et al., 2024), potentially initiating a data flywheel where growing human-written formal math data leads to more capable LLMs, which in turn eases the creation of more data.

Emerging opportunities have led to booming research activities in AI for formal mathematical reasoning. The number of publications in this field has almost doubled annually in 2023 and 2024 (Li et al., 2024b). AlphaProof leveraged formal reasoning to become the first AI to achieve the silver medal level in IMO. Developments in this field also have immediate applications in formal verification (Klein et al., 2009). While formal verification can lead to software and hardware systems that are exceedingly robust and secure, it has historically been too costly to deploy in all but the most safety-critical applications. AI can drastically reduce this cost by automating the formalization and proof effort needed to formally certify complex systems. This can lead to a future in which mass-produced software and hardware systems are far more robust than they are today.

We believe AI-based formal mathematical reasoning has reached an inflection point, with significant progress in the near future. This position paper maps out open challenges in data and algorithms and potential routes for future progress. It is not meant to be a comprehensive survey but to provide perspectives on where the field may go next and call on the community to unite to accelerate the progress.

## 2. AI4Math and the Formal Turn

After discussing the limitations of the informal approach, we introduce formal reasoning as a promising path.


Math-related web documents
Problems w/ step-by-step solutions
Problems w/ tool-integrated solutions
LLM pretrained on text and code
Base math LLM
Finetuned math LLM
Tool-integrated math LLM

**Problem**: Suppose that the sum of the squares of two complex numbers $x$ and $y$ is 7, and the sum of their cubes is 10. List all possible values for $x + y$, separated by commas.

**Solution**: Let's use `sympy` to calculate and print all possible values for $x + y$.

```
def possible_values():
    x, y = symbols("x y")
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1, eq2), (x, y))
    return [simplify(sol[0] + sol[1]) for sol in solutions]

print(possible_values())
```

```
>>> [-5, -5, 1, 1, 4, 4]
```

Removing duplicates, the possible values for $x + y$ are \boxed{-5, 1, 4}

*Figure 1.* A typical math LLM.

### 2.1. State-of-the-art Math LLMs and Their Limitations

**A Case Study of NuminaMath.** NuminaMath (Fleureau et al., 2024) won the first AIMO Progress Prize in July 2024 and is an excellent example of modern math LLMs, as it encompasses many key ingredients:

1. *Math pretraining* (Fig. 1 *Left*): The base math LLM results from continually pretraining a generic LLM on mathematical Web documents. NuminaMath adopted DeepSeekMath-Base 7B (Shao et al., 2024) as the base math LLM, which was trained on high-quality mathematical documents retrieved from Common Crawl through a carefully engineered data pipeline that combined automatic filtering and manual annotation.
2. *Finetuning on step-by-step solutions* (Fig. 1 *Middle*): To align the model with problem solving, one can finetune it on a carefully curated dataset of math problems with detailed, step-by-step solutions. NuminaMath collected 860K problems and solutions covering high school and competition math (Li et al., 2024a).
3. *Tool-integrated reasoning* (Fig. 1 *Right*): Finetuned math LLMs may still struggle with precise calculation (e.g., $162 \times 731$) and symbol manipulation. A simple solution is to outsource these operations to external tools such as SymPy (Meurer et al., 2017). NuminaMath performs tool-integrated reasoning that interleaves reasoning in natural language with tool invocation in Python. The key is, again, *data*. They follow ToRA (Gou et al., 2024b) and MuMath-Code (Yin et al., 2024) to collect a dataset of math problems with solutions in the form of natural language interleaved with tool invocation trajectories.

**Data Scarcity.** Training data plays a pivotal role throughout all ingredients of the informal approach, limiting its success to domains with abundant high-quality data, such as pre-college math. It is difficult to extend the approach to data-scarce domains such as advanced mathematics. Advanced mathematics is important for AI-driven scientific discovery, as it serves as the foundation of numerous scientific disciplines (e.g., climate modeling depends on partial differential equations). Moreover, the long-term goal of developing human-level AI mathematicians requires AI to handle novel aspects of mathematics. Novelty, by definition, implies difficulty in collecting in-distribution training data.

**Lack of Correctness Verifiability.** Another challenge lies in evaluation. Existing math LLMs are evaluated on benchmarks like MATH because many pre-college problems have numeric solutions that can be checked easily. For advanced mathematics, however, restricting to numeric solutions would deviate from common practice (Glazer et al., 2024), as it frequently deals with abstract conjectures and proofs. Verifying proofs can be a daunting task, even for experienced mathematicians (Klarreich, 2018). The situation becomes even more complicated when LLMs are used to generate proofs, as they are known to hallucinate plausibly.

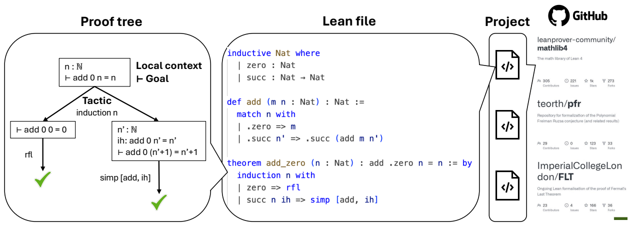
## 2.2. AI for Formal Mathematical Reasoning

**From Informal to Formal.** *Formal mathematics* addresses the challenges in data and evaluation, potentially enabling AI to tackle advanced mathematics. In this paper, it refers to mathematics grounded in formal systems, which have a syntax for well-formed formulas and can perform reasoning by manipulating formulas according to rules. Examples of formal systems include higher-order logic (Gordon, 2000) and dependent type theory (Martin-Löf & Sambin, 1984). They are used not only in math but to express computer programs and reason about semantics (Howard, 1980).

Formal systems are useful environments for AI to learn math. They guarantee the soundness of the reasoning, provide automatic feedback, and check if the goal has been achieved. This is crucial to addressing existing challenges in data scarcity and evaluation. Automatic feedback can serve as learning signals and alleviate the need for *human-created* training data. Rigorous proof verification can evaluate the model's reasoning without worrying about hallucination.

**Proof Assistants and Lean.** An important type of formal system is *proof assistants*. These are software tools that enable humans to write formal proofs about mathematics or verified software. Common examples of proof assistants include Coq (Barras et al., 1997), Isabelle (Nipkow et al., 2002), and Lean (Moura & Ullrich, 2021). They have different logical foundations but similar user interfaces. For simplicity, we will use Lean as an example.

Fig. 2 demonstrates how Lean is used to formalize mathematics. At its core, it is a programming language for writing not only conventional programs but also mathematical definitions, theorems, and proofs. Fig. 2 (*Middle*) is a Lean file. After defining natural numbers (`Nat`) and addition (`add`), it states and proves the theorem `add_zero` ($\forall n \in \mathbb{N}, 0 + n = n$). Lean can automatically check if the proof is correct with respect to the theorem statement.


Proof tree
n : ℕ
⊢ add 0 n = n
Local context
⊢ Goal
Tactic
induction n
⊢ add 0 0 = 0
n' : ℕ
ih: add 0 n' = n'
⊢ add 0 (n'+1) = n'+1
rfl
simp [add, ih]
Lean file
Project
GitHub
leanprover-community/mathlib4
The math library of Lean 4
teorth/pfr
Repository for formalization of the Polynomial Freiman Ruzsa conjecture (and related results)
ImperialCollegeLondon/FLT
Ongoing Lean formalisation of the proof of Fermat's Last Theorem

```
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat

def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')

theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]
```

*Figure 2.* Formalizing math using Lean (de Moura et al., 2015).

Theorem proving in Lean is an interactive process (Fig. 2 *Left*). It begins with the statement as the initial goal, and the user enters a proof step, known as a "tactic". Lean executes the tactic, transforming the goal into a list of subgoals. The user then inspects the new goals and enters new tactics, repeating this process until there are no goals left. This process implicitly defines a proof tree whose nodes are goals and edges are tactics. The user plays a key role here. While proof assistants like Lean were designed with human users in mind, in formal mathematical reasoning, the user can also be AI or human mathematicians in collaboration with AI (Buzzard, 2024; Tao, 2024).

Formalizing mathematics using Lean is like developing software (Fig. 2 *Right*). Files are organized into larger code units such as libraries and projects, which can be open-sourced on GitHub and reused by other projects. For example, the formalization of cutting-edge research often builds upon the basic concepts formalized in `mathlib` (Mathlib community, 2020): Lean's general-purpose mathematical library.

**AI Meets Formal Mathematics.** Integrating AI with proof assistants such as Lean can be mutually beneficial. Proof assistants provide data and environments for AI to learn math, whereas AI enhances the user experience of proof assistants, e.g., by automating simple proofs. Fig. 3 illustrates common tasks at this intersection: Given informal mathematics from textbooks or papers, *autoformalization* automatically translates it into formal theorems and proofs (Fig. 4). Given theorem statements, *theorem proving* aims to generate formal proofs. In addition to the statement, a theorem prover may have access to a large library of existing definitions and lemmas, such as `mathlib`, and can select useful definitions and lemmas from the library. Furthermore, AI for autoformalization and theorem proving can lead to new theorems and/or proofs that can enrich the library.

Fig. 5 is a common architecture for neural theorem provers, which combines tactic generation and proof search. Given the current goal, a neural network generates suggestions

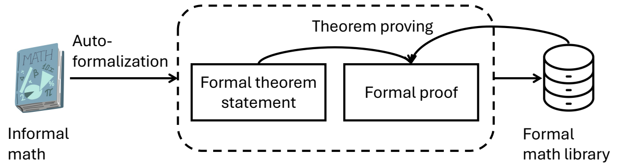
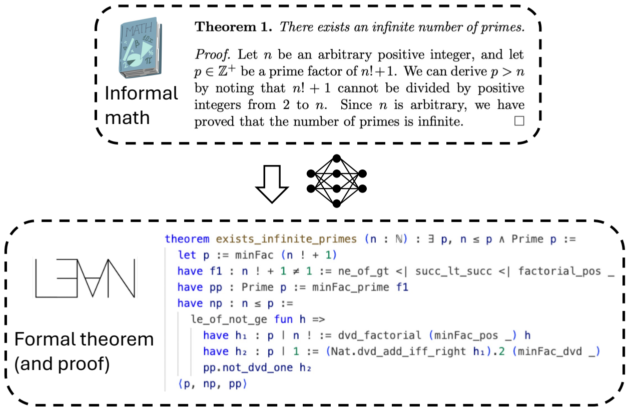


*Figure 3.* AI for formal mathematical reasoning in proof assistants.



**Theorem 1.** *There exists an infinite number of primes.*

*Proof.* Let $n$ be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n! + 1$ cannot be divided by positive integers from 2 to $n$. Since $n$ is arbitrary, we have proved that the number of primes is infinite. □

```
theorem exists_infinite_primes (n : ℕ) : ∃ p, n ≤ p ∧ Prime p :=
  let p := minFac (n ! + 1)
  have f1 : n ! + 1 ≠ 1 := ne_of_gt <| succ_lt_succ <| factorial_pos _
  have pp : Prime p := minFac_prime f1
  have np : n ≤ p :=
    le_of_not_ge fun h =>
      have h₁ : p ∣ n ! := dvd_factorial (minFac_pos _) h
      have h₂ : p ∣ 1 := (Nat.dvd_add_iff_right h₁).2 (minFac_dvd _)
      pp.not_dvd_one h₂
  ⟨p, np, pp⟩
```

*Figure 4.* Autoformalization: from informal to formal.

for the next tactic. The network is often trained on human-written proofs and can be finetuned using reinforcement learning. The generated tactics are assembled into a complete proof through repeated sampling or tree search.

**Synergies with Natural Language and System Design.** The formal and informal approaches are not mutually exclusive, nor should formal reasoning entirely supplant the informal. Instead, they can complement each other to enable complex reasoning that is both general and rigorous, e.g., integrating autoformalization with theorem proving to solve problems formulated in natural language (Zhou et al., 2024a). We refer to the combination of formal and informal reasoning as *verified reasoning in natural language*.

Moreover, formal mathematics has direct applications in the verification of software and hardware systems (Leroy, 2009). Here, one specifies the correctness/security requirements as formal statements and uses theorem proving to establish that the system satisfies its requirements. AI-driven autoformalization and theorem proving can potentially facilitate this process, significantly reducing the costs.

## 3. Recent Progress

We summarize recent progress in AI for formal mathematical reasoning. A more extensive discussion can be found in the extended version of this paper (Yang et al., 2024b).

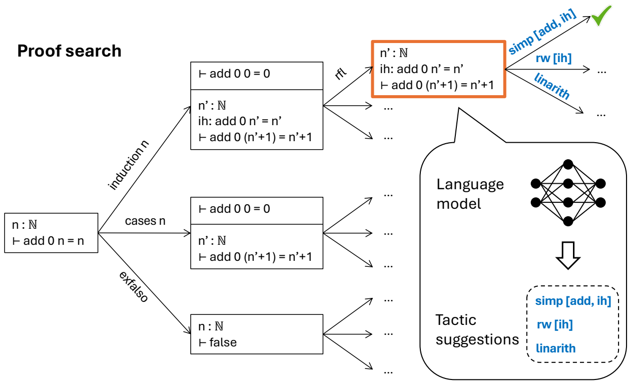


*Figure 5.* A common architecture for neural theorem proving.

**Autoformalization.** Early attempts of machine learning for autoformalization were hampered by the lack of aligned informal-formal corpora (Kaliszyk et al., 2017; Wang et al., 2018; 2020). The emergence of in-context learning in LLMs opens up a new paradigm, requiring only a few expert-constructed demonstrations (Wu et al., 2022). In addition, *in*formalization is generally easier than formalization, and we can use LLMs to perform back-translation, i.e., generating synthetic aligned corpora by auto-*in*formalizing existing formal statements (Jiang et al., 2024; Azerbayev et al., 2023). Finetuning a smaller model on this synthetic data led to notable improvements in autoformalization.

As the bridge between informal and formal mathematics, autoformalization has three immediate applications: (1) data augmentation for neural theorem provers (Wu et al., 2022; Xin et al., 2024; Wang et al., 2025; Ren et al., 2025), (2) guiding theorem proving via informal proofs (Jiang et al., 2023), and (3) validating informal reasoning (Zhou et al., 2024a; Olausson et al., 2023).

**Neural Theorem Proving.** Deep learning has been widely used for learning heuristics to find proofs in formal systems (Vaezipoor et al., 2021; Lederman et al., 2020). Holophrasm (Whalen, 2016) was the first system to demonstrate the feasibility of training deep neural networks to guide proof search. This paradigm was expanded in GPT-f (Polu & Sutskever, 2020), which trained a single Transformer to generate proof steps. It enjoyed substantial gains from math pretraining (Sec. 2.1). Subsequent approaches have trained richer architectures and exploited zero-shot prompting of LLMs. We highlight several prominent ideas:

- *Expert iteration* alternates between (1) using the prover to attempt unsolved problems and, (2) if new proofs are found, using them as training data to improve the prover. It leads to a performance gain that diminishes after a few iterations (Polu et al., 2023; Lample et al., 2022).

- *Learning from mistakes*: Formal proof environments can provide error messages when a proof step fails. COPRA (Thakur et al., 2024) included error messages in the prompts, utilizing LLMs' in-context learning capability to reduce the odds of repeating similar mistakes.
- *Informal proof sketches*: Formal theorem proving has also benefited from informal proofs. Draft, Sketch and Prove (DSP) (Jiang et al., 2023) used LLMs to generate a "proof sketch" in natural language, and then autoformalized it in Isabelle. Lean-STaR (Lin et al., 2024) interleaved formal and informal reasoning steps in theorem proving in Lean.
- *Premise selection* involves retrieving useful lemmas in proving a theorem (Kühlwein et al., 2012; Irving et al., 2016; Mikuła et al., 2024). ReProver (Yang et al., 2023) applied retrieval for neural theorem proving, where it first retrieved lemmas from a mathematical library. COPRA also used retrieved lemmas as part of their LLM prompts.

**Verified Reasoning in Natural Language.** Reasoning problems expressed in natural language may be difficult to completely formalize. In such cases, we still want *some* form of verification. Several works have used neural networks as verifiers (Lightman et al., 2024; Yang et al., 2022; Ling et al., 2024). While neural verifiers cannot formally guarantee the validity of the reasoning, they nonetheless provided a boost in overall performance and faithfulness.

Alternatively, one can combine LLM-based autoformalization with formal problem solving. SatLM (Ye et al., 2023) and LINC (Olausson et al., 2023) converted the entire problem into appropriate formats and called SAT/SMT solvers to produce solutions. LogicGuide (Poesia et al., 2024b) used a formal system to constrain the step-by-step deductions from the LLM, producing chain-of-thought reasoning that alternates between formal reasoning and natural language.

**Formal Verification and Verifiable Generation.** AI can automate many tedious aspects in theorem proving for system verification, e.g., generating initial proofs (Sanchez-Stern et al., 2020) and refining existing proofs (First et al., 2023). Moreover, it is useful in SMT-based verification tasks, including inferring loop invariants (Si et al., 2020) and generating helper assertions (Mugnier et al., 2024). Integrating AI and verification has been explored in the formal method community. For example, Seshia (2015) proposed SID (Structure, Induction, and Deduction), a general framework that combines inductive and deductive reasoning for verification. In SID, machine learning models can serve as the inductive engine to generate artifacts (e.g., loop invariants or proofs), while formal systems such as Lean serve as the deductive engine to answer queries about these artifacts.

A closely related challenge is to simultaneously generate code and formal proofs. LLM-generated code can be buggy and insecure (Pearce et al., 2022; Perry et al., 2023). Coupling generation with formal verification is a natural way to prevent such failures. One possibility is to first develop a formally verified program in Coq or Lean, with AI assistance, and then translate it into a more efficient implementation using compilers. This approach establishes a direct arc between theorem proving and generation. Another possibility is to incorporate LLM-based code and proof generation into a high-level verification-friendly language like Dafny (Misu et al., 2024) and Verus (Lattuada et al., 2023).

## 4. Open Challenges and Future Directions

Formal mathematical reasoning presents a wealth of challenging problems for AI. Here, we explore several open challenges and promising directions.

### 4.1. Data

Scaling the training data in formal mathematics is hampered by the scarcity of human-created formal proofs. The Proof Pile dataset (Azerbayev et al., 2024), which aggregated proofs from six different formal languages, collected only 500MB of formal proofs—orders of magnitude smaller than its informal counterparts. The issue is more pronounced in research mathematics, where even informal data is limited.

Researchers are exploring different strategies to overcome data scarcity. The first is autoformalization. We have a substantial amount of informal math data. Since formal proofs can be verified easily, if a system successfully formalizes even a small subset of the informal math data, it can self-improve through expert iteration, potentially covering an increasingly larger set with each iteration (Sec. 3). Another approach is synthetic data generation. For example, the training data in AlphaGeometry and TongGeometry consisted solely of synthetic geometry problems and solutions (Trinh et al., 2024; Zhang et al., 2024a). Mathematical axioms entail all provable facts, in principle containing *infinite* data. By generating synthetic data, AI can potentially explore and learn from the vast space of possible mathematics, at a scale that can drastically surpass the pace of human-created training data. If the method can be generalized, it would help in completely new mathematical domains, where even informal data might be scarce.

Autoformalization and synthetic data were combined in AlphaProof (Google DeepMind, 2024), which autoformalized one million IMO-like informal problems into one hundred million formal theorems, whose proofs were synthetically generated using expert iteration. It remains an open question to generalize this approach beyond domains where a large number of human-written problems are available, e.g., research mathematics. Those domains will likely require *conjecturing* new unseen statements (Poesia et al., 2024a).

Another promising strategy is knowledge transfer from different modalities. Specifically, code is closely related to mathematics, as both require symbolic reasoning. This similarity has been exploited to improve AI's general mathematical capabilities (Gao et al., 2023; Guo et al., 2024; Dubey et al., 2024), though it is still an open question how to leverage data-rich programming languages such as Python to enhance reasoning in *formal* mathematical languages.

### 4.2. Algorithms

**Autoformalization at Scale.** A major bottleneck in autoformalization is evaluating whether the autoformalized statement is logically equivalent to the ground truth. Proxy metrics such as BLEU (Papineni et al., 2002) do not correlate well with human judgment (Jiang et al., 2024), but relying on humans is not scalable. Possible ideas for better automated metrics include checking logical equivalence via automated provers (Murphy et al., 2024; Li et al., 2024c).

Autoformalization goes beyond translation, as some problems (e.g., IMO 2024 P5) may require complex reasoning, retrieving existing definitions, or even inventing new ones. For such problems, it is natural to break down the reasoning process into smaller steps, e.g., retrieving definitions before formalizing the statement or generating high-level sketches before formalizing the proof. We anticipate benefits from smaller steps and process supervision (Lightman et al., 2024; Lu et al., 2024) and call for autoformalization to be more interactive (Szegedy, 2020).

**Proof Search and Test-Time Compute.** Search is fundamental in many reasoning systems. Many neural theorem provers combine tactic generation with proof search (Fig. 5). The search strategy ranges from independent sampling of multiple candidates to sophisticated algorithms like Monte Carlo Tree Search (Lample et al., 2022). Scaling search to exploit vast test-time compute has emerged as a promising approach for both formal and informal reasoning (Google DeepMind, 2024; Zhang et al., 2024b; Xie et al., 2024b).

Many myths and trade-offs surrounding proof search remain unexplored. Is proof search really necessary, given that generating complete proofs can achieve lower latency (First et al., 2023; Xin et al., 2024). For a fixed compute budget, should we use smaller models with more search steps, or larger models (Wu et al., 2024a; Blaauwbroek et al., 2024)? How do different search algorithms compare? To answer these questions and guide the development of future provers, we need a systematic evaluation of existing methods. Such an evaluation is lacking due to the inherent challenge in evaluating theorem proving in a fair and unified manner. It is unclear how to compare provers targeting different proof assistants. Even within the same proof assistant, a prover's performance is multifaceted and depends on resource constraints (e.g., hardware and time limits), making it difficult to consolidate performance into a single metric. A comprehensive evaluation that addresses these complexities would be immensely valuable. Despite these challenges, researchers are exploring various directions to improve proof search, such as developing value models to assess the promise of proof goals (Lample et al., 2022; Wu et al., 2024b).

Proof search alone does not solve theorem proving, where a fundamental challenge is a discrete, infinite action space. Proof search cannot succeed if the model cannot produce high-quality actions in the first place. In the context of theorem proving, mathematical creativity can manifest as actions exceeding the current model's capabilities, akin to the "divine move (神之一手)"—a legendary concept in Go. We would not expect to find them if the action space were an infinite, unstructured list. Fortunately, mathematics is structured, making it possible—though still challenging—to find the divine moves (Gowers, 2022). Next, we discuss several ways of leveraging structures in mathematics.

**Exploiting Hierarchies and Learning Abstractions.** Theorems are built upon smaller lemmas, which in turn break down into even simpler subgoals. Several existing theorem provers leverage this hierarchical structure. Draft, Sketch, and Prove (Jiang et al., 2023) transformed informal proofs into formal "proof sketches"—skeletons of formal proofs with "holes", i.e., open goals left unproven, yielding a hierarchical structure. POETRY (Wang et al., 2024) recursively decomposed goals in proof sketches using an LLM. While these works demonstrated the potential of hierarchical decomposition, it is still a significant challenge to decompose realistic high-level goals with current LLMs.

Abstraction is central to human mathematical practice. We first learn natural numbers through counting; years later, those operations show up in solving equations but do not require as much attention anymore. In interactive theorem proving, abstractions can be encapsulated in new definitions, lemmas, and tactics. While most existing methods assume they are predefined and fixed, recent work has explored learning abstractions. For instance, LEGO-Prover (Xin et al., 2023) used LLMs to propose and prove new lemmas, integrating them into its library to help prove further theorems. Lemma mining from existing proof corpora has also been explored (Kaliszyk & Urban, 2013; Zhou et al., 2024b). These lemmas, not explicitly factored out by humans, are still useful for automation. On learning *tactics*, Peano (Poesia & Goodman, 2023) and LEMMA (Li et al., 2022) have proposed to learn simple proof strategies from an agent's own solutions to past problems, in a bootstrapping fashion. However, these approaches have so far been limited to simpler formal systems, and it is still an open challenge to synthesize entirely new tactics in full-fledged formal theorem proving languages like Lean.

**Incorporating External Knowledge.** Formal mathematical reasoning can benefit from explicitly retrieving and incorporating knowledge from databases of existing lemmas/definitions. ReProver (Yang et al., 2023) and COPRA (Thakur et al., 2024) demonstrated performance gains through standard retrievers like BM25 (Robertson et al., 2009) and Dense Passage Retrieval (Karpukhin et al., 2020). A promising direction involves developing retrieval methods tailored to formal math, e.g., structured or neurosymbolic retrievers. Another avenue is to grow the knowledge base dynamically. For example, the system could decompose high-level proof goals into subgoals, cache a subset of these subgoals as modules, and use them in subsequent proof efforts. Deciding which subgoals are "interesting" enough to be modularized in this way is a challenge.

### 4.3. Formal Verification and Verifiable Generation

Like AI4Math, we envision a growing need for formal reasoning in AI-based software and hardware generation, with assurance of correctness and security. While syntactical correctness can be guaranteed by constrained decoding (Beurer-Kellner et al., 2024), ensuring semantic properties, such as those validated by compilers, remains a challenge. Moreover, formal reasoning can help programmers understand AI-generated code (Ferdowsi et al., 2024).

Formal verification poses unique challenges. For example, a necessary but challenging step is encoding the target system and the correctness requirements in the proof assistant, similar to formalizing mathematics. However, while mathematical statements tend to assert properties of established mathematical objects, statements in formal verification typically concern bespoke procedures and datatypes. Also, proofs tend to be more repetitive and heavy on case-splits and inductive reasoning about recursive functions and datatypes. Finally, unlike statements in mathematics research, real-world software and hardware systems are characterized by large codebases and frequent changes. For instance, seL4 (Klein et al., 2009) consists of about 200,000 lines of specifications and proofs in Isabelle. Verifying these systems requires not only theorem proving but also rigorous management of specifications and proofs—an exciting yet underexplored direction for AI (Ringer et al., 2019).

It is natural to couple formal verification and AI-based generation to simultaneously generate code, formal specifications (i.e., pre/post-conditions, loop invariants, and helper assertions), and proofs. Then a program verifier or a theorem prover can check if the code is consistent with the specifications and proofs. This approach has been explored in recent research (Sun et al., 2024a; Yang et al., 2024a) and can potentially reduce verification efforts and enhance software and hardware reliability. However, a key challenge is to ensure the trustworthiness of the generated specifications—that they accurately reflect developers' intent.

## 5. Milestones and Success Measures

Inspired by the levels of automation for self-driving cars (SAE, 2024), we propose levels for AI-based formal mathematical reasoning. A more extensive discussion can be found in the extended version of our paper (Yang et al., 2024b).

### 5.1. Autoformalization

- *Level 0, representing knowledge in formal systems to support manual formalization*: Achieved by modern proof assistants such as Lean.
- *Level 1, generating autoformalization candidates and collecting human feedback*: LLMs can often generate good autoformalization candidates, but we need a system to gather and store human feedback, e.g., bug fixes and revisions made by humans to make the candidates usable. The system would also keep informal-formal pairs synchronized as the formal statements continue to develop.
- *Level 2, robust and faithful translations between informal and formal*: Model performance at this level could be assessed using human-curated benchmarks, including challenges from the ICML 2024 Math-AI workshop (mat, 2024; Huang et al., 2024b), ProofNet (Azerbayev et al., 2023), Herald (Gao et al., 2025), and Con-NF (Liu et al., 2025). However, a major obstacle is how to automatically evaluate autoformalized statements (Sec. 4.2).
- *Level 3, inferring missing information and flagging situations when a gap cannot be filled*: Implicit assumptions and hand-waived proof steps frequently pose challenges in formalizing mathematics. Bridging these information gaps requires robust reasoning capabilities: Proof gaps may be filled by neural or symbolic theorem provers, while missing assumptions can be resolved using abductive reasoning or counterexamples (Bundy et al., 2005; Blanchette & Nipkow, 2010). The main challenge is for the models to identify gaps—such as assessing the likelihood that a statement is provable or can be adjusted—even when it cannot bridge the gap immediately.
- *Level 4, self-correcting erroneous or inconsistent inputs by understanding human intentions*: At this stage, the autoformalization model focuses more on capturing human intentions and may rely on its own self-consistency to eliminate errors. Advancements here will be closely linked to natural language reasoning (Sec. 5.3).
- *Level 5, proposing novel definitions that can reduce proof complexity*: AI can serve as "theory builders" that reshape the proving process through better abstraction or concept formulation. For instance, filters (i.e., a set of sets satisfying certain properties) are rarely taught in standard math

curricula but have become convenient for formalizing limits in various proof assistants. Automatically devising definitions like filters is what we hope AI can achieve.

## 5.2. Theorem Proving

- *Level 0, checking formal proofs*: Achieved by modern proof assistants such as Lean.
- *Level 1, assisting humans to develop formal proofs by suggesting definitions, lemmas, proof steps, etc.*: Library search engines like LeanSearch (Gao et al., 2024) and proof completion "copilots" (Dohmke, 2023; Song et al., 2024) can be highly helpful, though humans are still responsible for the main job of developing the proof.
- *Level 2, human-implemented tactics for proof automation*: These are domain-specific procedures for automating certain classes of proofs, e.g., `omega` and `nlinarith` can automatically solve many equalities and inequalities. This level involves no machine learning but mostly human-engineered domain-specific methods to produce proofs.
- *Level 3, proving simple theorems automatically in a domain-general fashion*: Recent neural theorem provers are domain-agnostic and evaluated on benchmarks targeting this level, e.g., CoqGym (Yang & Deng, 2019), LeanDojo (Yang et al., 2023), MiniF2F (Zheng et al., 2022), and PutnamBench (Tsoukalas et al., 2024). These systems are limited to relatively simple proofs, typically not the most time-consuming ones, but they can still be useful for closing simple gaps in larger proofs.
- *Level 4, contributing to formalization projects autonomously*: Beyond proving individual theorems, AI should be able to break down larger results, state new definitions and lemmas, and potentially explore different alternatives as the project develops. Evaluation may require new benchmarks constructed from GitHub metadata, such as issues and commits, of real-world formalizations.
- *Level 5, solving problems and discovering new math beyond the human level*: Out of reach for current AI systems. One challenge will be to measure progress meaningfully towards this open-ended goal, since our current evaluations only test knowledge of existing mathematics.

## 5.3. Verified Reasoning in Natural Language

- *Level 0, stepwise natural language reasoning w/o verification*: Chain-of-thought (Wei et al., 2022) is effective but its reasoning can be brittle, incorrect, or unfaithful (Shi et al., 2023; Lanham et al., 2023; Ling et al., 2024).
- *Level 1, stepwise natural language reasoning with neural verification*: Neural verifiers can improve reasoning, e.g., by selecting the best output from many generated candidates (Cobbe et al., 2021). Evaluation can be done using standard benchmarks like MATH (Hendrycks et al., 2021) or by directly measuring whether the model can identify reasoning errors (Hong et al., 2024; Zheng et al., 2024a). Standard benchmarks like MATH suffer from data contamination (Dong et al., 2024). To mitigate the issue, dynamically generated benchmarks such as GSM-Symbolic (Mirzadeh et al., 2024) or private benchmarks such as FrontierMath (Glazer et al., 2024) may be useful.
- *Level 2, tool-integrated reasoning using SymPy, NumPy, etc.*: Models can leverage external tools to perform computation that neural networks struggle to learn reliably, e.g., numerical calculations and symbol manipulation. They can be evaluated on math problems requiring intricate computations, e.g., the MATH dataset or AIME.
- *Level 3, Reasoning seamlessly in natural language and formal systems such as Lean*: Instead of formalizing the entire problem (Zhou et al., 2024a), the model can interleave natural language with formal reasoning, selectively determining which parts of reasoning to process using formal systems. Benchmarks should evaluate not only the final answer but also the quality of reasoning, and they should include math problems that resist complete formalization, e.g., IMO as taken by human contestants, without manual formalization as AlphaProof did.
- *Level 4, complex mathematical reasoning and planning in real-world applications*: Applications of complex reasoning often contain mathematical components along with other components such as commonsense and human preferences. In scenarios like travel planning (Xie et al., 2024a) or calendar scheduling (Zheng et al., 2024b), AI could formulate the task as a constraint satisfaction problem and solve it using appropriate solvers. Achieving this capability would enable a wide range of applications.

## 5.4. Formal Verification and Verifiable Generation

- *Level 0, code generation without verification.*
- *Level 1, verifying and synthesizing small programs with simple properties*: Several benchmarks have targeted *verification* at this level (Loughridge et al., 2024; Lohn & Welleck, 2024; Aggarwal et al., 2024), but current models still fall short. We are not aware of large-scale benchmarks for synthesizing verified code together with a formal specification, even at Level 0 (Brandfonbrener et al. (2024) introduced a small-scale benchmark in this direction).
- *Level 2, verifying and synthesizing entire projects with complex functional and security properties*: This level requires decomposing large systems into smaller verifiable components, a task currently performed by humans (Gu et al., 2016) but may be tackled by advanced AI agents capable of planning and problem-solving to navigate the intricate dependencies and interactions in large codebases. Benchmarks should incorporate project-level

context, e.g., extracted from existing verification projects systems (Leroy et al., 2016; Zhang et al., 2024c).

- *Level 3, proof and system maintenance*: System designs and implementations constantly evolve, and so must their proofs to stay synced. AI should provide assistance when developers update the system or refactor proofs (Ringer, 2021). To evaluate this capability level, benchmarks can be constructed from the change history of verified systems (Reichel et al., 2023). These benchmarks should capture a variety of scenarios, including minor bug fixes, major feature additions, and comprehensive refactoring.
- *Level 4, helping users generate, explain, and debug formal specifications*: AI should aid users in writing specifications, which requires abstracting and converting user requirements into formal languages. The evaluation can leverage verified codebases. Instead of generating proofs and code given the specifications, we treat them as ground truth and use them to evaluate specifications generated by the model. The system should be interactive to engage with users, offering suggestions and clarifications.

## 6. Alternative Views

We argue that formal mathematical reasoning is essential to address data scarcity and lack of verifiability in the informal approach (Sec. 2.1). **An alternative view is that these challenges could be addressed by neural networks alone, without formal systems.** For example, OpenAI o1 and DeepSeek-R1 have demonstrated impressive capabilities on problems with numeric answers. However, it remains unclear if they can solve problems requiring rigorous intermediate steps, such as theorem proving. Neural verifiers could add rigor to the informal approach, but their success hinges on whether neural networks can achieve (1) self-verification/correction and (2) easy-to-hard generalization (Sun et al., 2024b). Despite extensive studies on self-verification, current LLMs struggle with *intrinsic self-verification*—verifying their own generations without external feedback (Huang et al., 2024a; Stechly et al., 2024; Gou et al., 2024a; Zheng et al., 2024a; Gu et al., 2024).

## 7. Conclusion

We advocated for formal mathematical reasoning as an important complement to the informal approach, highlighting its potential to advance AI in math and verified system design. We hope to present coherent perspectives that unite previously fragmented efforts across different fields, fostering discussion, community building, and a future roadmap.

## Acknowledgements

We gratefully acknowledge Jeremy Avigad, Albert Q. Jiang, Zhaoyu Li, Peter O'Hearn, Daniel Selsam, Sanjit A. Seshia, Armando Solar-Lezama, and Terence Tao for providing valuable feedback on an initial version of this paper.

## Impact Statement

This position paper aims to advance AI for mathematical reasoning and its application in verifiable software and hardware system design. There are many potential societal consequences of these areas, none of which we feel must be specifically highlighted here.

## References

ICML 2024 Challenges on Automated Math Reasoning. https://sites.google.com/view/ai4mathworkshopicml2024/challenges, 2024.

Aggarwal, P., Parno, B., and Welleck, S. AlphaVerus: Bootstrapping formally verified code generation through self-improving translation and treefinement. *arXiv preprint arXiv:2412.06176*, 2024.

Aniva, L., Sun, C., Miranda, B., Barrett, C., and Koyejo, S. Pantograph: A machine-to-machine interaction interface for advanced theorem proving, high level reasoning, and data extraction in Lean 4. *arXiv preprint arXiv:2410.16429*, 2024.

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Azerbayev, Z., Schoelkopf, H., Paster, K., Dos Santos, M., McAleer, S. M., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*, 2024.

Barras, B., Boutin, S., Cornes, C., Courant, J., Filliatre, J.-C., Gimenez, E., Herbelin, H., Huet, G., Munoz, C., Murthy, C., et al. *The Coq proof assistant reference manual: Version 6.1*. PhD thesis, Inria, 1997.

Beurer-Kellner, L., Fischer, M., and Vechev, M. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *International Conference on Machine Learning (ICML)*, 2024.

Blaauwbroek, L., Olšák, M., Rute, J., Massolo, F. I. S., Piepenbrock, J., and Pestun, V. Graph2Tac: Online representation learning of formal math concepts. In *International Conference on Machine Learning (ICML)*, 2024.

Blanchette, J. C. and Nipkow, T. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *International Conference on Interactive Theorem Proving (ITP)*, 2010.

Brandfonbrener, D., Henniger, S., Raja, S., Prasad, T., Loughridge, C. R., Cassano, F., Hu, S. R., Yang, J., Byrd, W. E., Zinkov, R., and Amin, N. VerMCTS: Synthesizing multi-step programs using a verifier, a large language model, and tree search. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=HmB9uZTzaD.

Bundy, A., Jamnik, M., and Fugard, A. What is a proof? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2005.

Buzzard, K. Can AI do maths yet? thoughts from a mathematician. https://xenaproject.wordpress.com/2024/12/22/can-ai-do-maths-yet-thoughts-from-a-mathematician/, 2024.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

de Moura, L., Kong, S., Avigad, J., Van Doorn, F., and von Raumer, J. The Lean theorem prover (system description). In *International Conference on Automated Deduction (CADE)*, 2015.

Dohmke, T. GitHub Copilot X: The AI-powered developer experience. https://github.blog/news-insights/product-news/github-copilot-x-the-ai-powered-developer-experience, 2023.

Dong, Y., Jiang, X., Liu, H., Jin, Z., and Li, G. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics: ACL*, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ferdowsi, K., Huang, R., James, M. B., Polikarpova, N., and Lerner, S. Validating AI-generated code with live programming. In *Conference on Human Factors in Computing Systems (CHI)*, 2024.

First, E., Rabe, M. N., Ringer, T., and Brun, Y. Baldur: Whole-proof generation and repair with large language models. In *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2023.

Fleureau, Y., Li, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S. C., and Rasul, K. How NuminaMath won the 1st AIMO Progress Prize. https://huggingface.co/blog/winning-aimo-progress-prize, 2024.

Gao, G., Ju, H., Jiang, J., Qin, Z., and Dong, B. A semantic search engine for Mathlib4. *arXiv preprint arXiv:2403.13310*, 2024.

Gao, G., Wang, Y., Jiang, J., Gao, Q., Qin, Z., Xu, T., and Dong, B. Herald: A natural language annotated Lean 4 dataset. In *International Conference on Learning Representations (ICLR)*, 2025.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023.

Gauthier, T., Kaliszyk, C., Urban, J., Kumar, R., and Norrish, M. TacticToe: learning to prove with tactics. *Journal of Automated Reasoning*, 2021.

Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., Santos, E. d. O., et al. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024.

Google DeepMind. AI achieves silver-medal standard solving international mathematical olympiad problems. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024.

Gordon, M. From LCF to HOL: a short history. 2000.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024a.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Huang, M., Duan, N., and Chen, W. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations (ICLR)*, 2024b.

Gowers, T. How can it be feasible to find proofs? https://drive.google.com/file/d/1-FFa6nMVg18m1zPtoAQrFalwpx2YaGK4/view, 2022.

Gu, A., Li, W.-D., Jain, N., Olausson, T. X., Lee, C., Sen, K., and Solar-Lezama, A. The counterfeit conundrum: Can code language models grasp the nuances of their incorrect generations? In *Findings of the Association for Computational Linguistics: ACL*, 2024.

Gu, R., Shao, Z., Chen, H., Wu, X. N., Kim, J., Sjöberg, V., and Costanzo, D. CertiKOS: An extensible architecture for building certified concurrent OS kernels. In *Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y. K., Luo, F., Xiong, Y., and Liang, W. DeepSeek-Coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.

Hong, R., Zhang, H., Pang, X., Yu, D., and Zhang, C. A closer look at the self-verification abilities of large language models in logical reasoning. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Howard, W. A. The formulae-as-types notion of construction. *To HB Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 1980.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024a.

Huang, Y., Lin, X., Liu, Z., Cao, Q., Xin, H., Wang, H., Li, Z., Song, L., and Liang, X. Mustard: Mastering uniform synthesis of theorem and proof data. In *International Conference on Learning Representations (ICLR)*, 2024b.

Irving, G., Szegedy, C., Alemi, A. A., Eén, N., Chollet, F., and Urban, J. DeepMath - deep sequence models for premise selection. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Jiang, A. Q., Welleck, S., Zhou, J. P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., and Wu, Y. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations (ICLR)*, 2023.

Jiang, A. Q., Li, W., and Jamnik, M. Multi-language diversity benefits autoformalization, 2024.

Kaliszyk, C. and Urban, J. Lemma mining over HOL Light. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, 2013.

Kaliszyk, C., Urban, J., and Vyskocil, J. System description: statistical parsing of informalized Mizar formulas. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2017.

Kaliszyk, C., Urban, J., Michalewski, H., and Olšák, M. Reinforcement learning of theorem proving. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Klarreich, E. Titans of mathematics clash over epic proof of ABC conjecture. https://www.quantamagazine.org/titans-of-mathematics-clash-over-epic-proof-of-abc-conjecture-20180920/, 2018.

Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., and Winwood, S. seL4: Formal verification of an OS kernel. In *Symposium on Operating systems principles (SOSP)*, 2009.

Kühlwein, D., van Laarhoven, T., Tsivtsivadze, E., Urban, J., and Heskes, T. Overview and evaluation of premise selection techniques for large theory mathematics. In *Automated Reasoning: 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings 6*, pp. 378–392. Springer, 2012.

Lample, G., Lacroix, T., Lachaux, M.-A., Rodriguez, A., Hayat, A., Lavril, T., Ebner, G., and Martinet, X. HyperTree proof search for neural theorem proving. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Lattuada, A., Hance, T., Cho, C., Brun, M., Subasinghe, I., Zhou, Y., Howell, J., Parno, B., and Hawblitzel, C. Verus: Verifying rust programs using linear ghost types. *Proceedings of the ACM on Programming Languages*, 2023.

Lederman, G., Rabe, M., Seshia, S., and Lee, E. A. Learning heuristics for quantified boolean formulas through reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Leino, K. R. M. Dafny: An automatic program verifier for functional correctness. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, 2010.

Leroy, X. Formal verification of a realistic compiler. *Communications of the ACM*, 2009.

Leroy, X., Blazy, S., Kästner, D., Schommer, B., Pister, M., and Ferdinand, C. CompCert-a formally verified optimizing compiler. In *Embedded Real Time Software and Systems (ERTS)*, 2016.

Li, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S., Rasul, K., Yu, L., Jiang, A. Q., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., and Polu, S. NuminaMath: The largest public dataset in AI4Maths with 860k pairs of competition math problems and solutions. https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf, 2024a.

Li, Z., Poesia, G., Costilla-Reyes, O., Goodman, N., and Solar-Lezama, A. Lemma: Bootstrapping high-level mathematical reasoning with learned symbolic abstractions. *arXiv preprint arXiv:2211.08671*, 2022.

Li, Z., Sun, J., Murphy, L., Su, Q., Li, Z., Zhang, X., Yang, K., and Si, X. A survey on deep learning for theorem proving. In *Conference on Language Modeling (COLM)*, 2024b.

Li, Z., Wu, Y., Li, Z., Wei, X., Zhang, X., Yang, F., and Ma, X. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *Neural Information Processing Systems (NeurIPS)*, 2024c.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Lin, H., Sun, Z., Yang, Y., and Welleck, S. Lean-STaR: Learning to interleave thinking and proving. *arXiv preprint arXiv:2407.10040*, 2024.

Ling, Z., Fang, Y., Li, X., Huang, Z., Lee, M., Memisevic, R., and Su, H. Deductive verification of chain-of-thought reasoning. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Liu, Q., Zheng, X., Lu, X., Cao, Q., and Yan, J. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *International Conference on Learning Representations (ICLR)*, 2025.

Lohn, E. and Welleck, S. miniCodeProps: a minimal benchmark for proving code properties. *arXiv preprint arXiv:2406.11915*, 2024.

Loughridge, C., Sun, Q., Ahrenbach, S., Cassano, F., Sun, C., Sheng, Y., Mudide, A., Misu, M. R. H., Amin, N., and Tegmark, M. DafnyBench: A benchmark for formal software verification. *arXiv preprint arXiv:2406.08467*, 2024.

Lu, J., Wan, Y., Liu, Z., Huang, Y., Xiong, J., Liu, C., Shen, J., Jin, H., Zhang, J., Wang, H., Yang, Z., Tang, J., and Guo, Z. Process-driven autoformalization in Lean 4. *arXiv preprint arXiv:2406.01940*, 2024.

Martin-Löf, P. and Sambin, G. *Intuitionistic type theory*. Bibliopolis Naples, 1984.

Mathlib community. The Lean mathematical library. In *Certified Programs and Proofs (CPP)*, 2020.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 2017.

Mikuła, M., Tworkowski, S., Antoniak, S., Piotrowski, B., Jiang, A. Q., Zhou, J. P., Szegedy, C., Kuciński, Ł., Miłoś, P., and Wu, Y. Magnushammer: A transformer-based approach to premise selection. In *International Conference on Learning Representations (ICLR)*, 2024.

Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Misu, M. R. H., Lopes, C. V., Ma, I., and Noble, J. Towards AI-assisted synthesis of verified Dafny methods. In *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2024.

Moura, L. d. and Ullrich, S. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction (CADE)*, 2021.

Mugnier, E., Gonzalez, E. A., Jhala, R., Polikarpova, N., and Zhou, Y. Laurel: Generating Dafny assertions using large language models. *arXiv preprint arXiv:2405.16792*, 2024.

Murphy, L., Yang, K., Sun, J., Li, Z., Anandkumar, A., and Si, X. Autoformalizing Euclidean geometry. In *International Conference on Machine Learning (ICML)*, 2024.

Newell, A. and Simon, H. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 1956.

Nipkow, T., Wenzel, M., and Paulson, L. C. *Isabelle/HOL: a proof assistant for higher-order logic*. 2002.

Olausson, T., Gu, A., Lipkin, B., Zhang, C., Solar-Lezama, A., Tenenbaum, J., and Levy, R. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

OpenAI. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, 2024.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., and Karri, R. Asleep at the keyboard? assessing the security of Github Copilot's code contributions. In *Symposium on Security and Privacy*, 2022.

Perry, N., Srivastava, M., Kumar, D., and Boneh, D. Do users write more insecure code with AI assistants? In *Conference on Computer and Communications Security (CCS)*, 2023.

Poesia, G. and Goodman, N. D. Peano: learning formal mathematical reasoning. *Philosophical Transactions of the Royal Society A*, 2023.

Poesia, G., Broman, D., Haber, N., and Goodman, N. D. Learning formal mathematics from intrinsic motivation. In *Neural Information Processing Systems (NeurIPS)*, 2024a.

Poesia, G., Gandhi, K., Zelikman, E., and Goodman, N. D. Certified deductive reasoning with language models. *Transactions on Machine Learning Research (TMLR)*, 2024b.

Polu, S. and Sutskever, I. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Polu, S., Han, J. M., Zheng, K., Baksys, M., Babuschkin, I., and Sutskever, I. Formal mathematics statement curriculum learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Reichel, T., Henderson, R., Touchet, A., Gardner, A., and Ringer, T. Proof repair infrastructure for supervised models: Building a large proof repair dataset. In *International Conference on Interactive Theorem Proving (ITP)*, 2023.

Ren, Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., et al. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Ringer, T. *Proof Repair*. University of Washington, 2021.

Ringer, T., Palmskog, K., Sergey, I., Gligoric, M., and Tatlock, Z. QED at large: A survey of engineering of formally verified software. *Foundations and Trends® in Programming Languages*, 2019.

Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 2009.

SAE. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. https://www.sae.org/standards/content/j3016_202104/, 2024.

Sanchez-Stern, A., Alhessi, Y., Saul, L., and Lerner, S. Generating correctness proofs with neural networks. In *SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2020.

Seshia, S. A. Combining induction, deduction, and structure for verification and synthesis. *Proceedings of the IEEE*, 2015.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., and Zhou, D. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*, 2023.

Si, X., Naik, A., Dai, H., Naik, M., and Song, L. Code2Inv: A deep learning framework for program verification. In *International Conference on Computer Aided Verification (CAV)*, 2020.

Song, P., Yang, K., and Anandkumar, A. Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv: Arxiv-2404.12534*, 2024.

Stechly, K., Valmeekam, K., and Kambhampati, S. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*, 2024.

Sun, C., Sheng, Y., Padon, O., and Barrett, C. Clover: Closed-loop verifiable code generation. In *International Symposium on AI Verification*, 2024a.

Sun, Z., Yu, L., Shen, Y., Liu, W., Yang, Y., Welleck, S., and Gan, C. Easy-to-hard generalization: Scalable alignment beyond human supervision. In *Neural Information Processing Systems (NeurIPS)*, 2024b.

Szegedy, C. A promising path towards autoformalization and general artificial intelligence. In *International Conference on Intelligent Computer Mathematics (CICM)*, 2020.

Tao, T. The potential for AI in science and mathematics. https://www.youtube.com/watch?v=_sTDSO74D8Q, 2024.

Thakur, A., Tsoukalas, G., Wen, Y., Xin, J., and Chaudhuri, S. An in-context learning agent for formal theorem-proving. In *Conference on Language Modeling (COLM)*, 2024.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 2024.

Tsoukalas, G., Lee, J., Jennings, J., Xin, J., Ding, M., Jennings, M., Thakur, A., and Chaudhuri, S. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Vaezipoor, P., Lederman, G., Wu, Y., Maddison, C., Grosse, R. B., Seshia, S. A., and Bacchus, F. Learning branching heuristics for propositional model counting. In *AAAI Conference on Artificial Intelligence*, 2021.

Wang, H., Xin, H., Liu, Z., Li, W., Huang, Y., Lu, J., Yang, Z., Tang, J., Yin, J., Li, Z., and Liang, X. Proving theorems recursively. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Wang, H., Unsal, M., Lin, X., Baksys, M., Liu, J., Santos, M. D., Sung, F., Vinyes, M., Ying, Z., Zhu, Z., et al. Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.

Wang, Q., Kaliszyk, C., and Urban, J. First experiments with neural translation of informal to formal mathematics. In *International Conference on Intelligent Computer Mathematics (CICM)*, 2018.

Wang, Q., Brown, C., Kaliszyk, C., and Urban, J. Exploration of neural machine translation in autoformalization of mathematics in Mizar. In *Certified Programs and Proofs (CPP)*, 2020.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Whalen, D. Holophrasm: a neural automated theorem prover for higher-order logic. *arXiv preprint arXiv:1608.02644*, 2016.

Wu, Y., Jiang, A., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024a.

Wu, Z., Huang, S., Zhou, Z., Ying, H., Wang, J., Lin, D., and Chen, K. InternLM2.5-StepProver: Advancing automated theorem proving via expert iteration on large-scale Lean problems. *arXiv preprint arXiv:2410.15700*, 2024b.

Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., Xiao, Y., and Su, Y. TravelPlanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning (ICML)*, 2024a.

Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. Monte Carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024b.

Xin, H., Wang, H., Zheng, C., Li, L., Liu, Z., Cao, Q., Huang, Y., Xiong, J., Shi, H., Xie, E., et al. LEGO-Prover: Neural theorem proving with growing libraries. In *International Conference on Learning Representations (ICLR)*, 2023.

Xin, H., Guo, D., Shao, Z., Ren, Z., Zhu, Q., Liu, B., Ruan, C., Li, W., and Liang, X. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

Yang, C., Li, X., Misu, M. R. H., Yao, J., Cui, W., Gong, Y., Hawblitzel, C., Lahiri, S., Lorch, J. R., Lu, S., et al. AutoVerus: Automated proof generation for Rust code. *arXiv preprint arXiv:2409.13082*, 2024a.

Yang, K. and Deng, J. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning (ICML)*, 2019.

Yang, K., Deng, J., and Chen, D. Generating natural language proofs with verifier-guided search. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Yang, K., Poesia, G., He, J., Li, W., Lauter, K., Chaudhuri, S., and Song, D. Formal mathematical reasoning: A new frontier in AI. *arXiv preprint arXiv:2412.16075*, 2024b.

Ye, X., Chen, Q., Dillig, I., and Durrett, G. SatLM: Satisfiability-aided language models using declarative prompting. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Yin, S., You, W., Ji, Z., Zhong, G., and Bai, J. MuMath-Code: Combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning. *arXiv preprint arXiv:2405.07551*, 2024.

Zhang, C., Song, J., Li, S., Liang, Y., Ma, Y., Wang, W., Zhu, Y., and Zhu, S.-C. Proposing and solving olympiad geometry with guided tree search. *arXiv preprint arXiv:2412.10673*, 2024a.

Zhang, D., Zhoubian, S., Yue, Y., Dong, Y., and Tang, J. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Neural Information Processing Systems (NeurIPS)*, 2024b.

Zhang, L., Lu, S., and Duan, N. Selene: Pioneering automated proof in software verification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024c.

Zheng, C., Zhang, Z., Zhang, B., Lin, R., Lu, K., Yu, B., Liu, D., Zhou, J., and Lin, J. ProcessBench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024a.

Zheng, H. S., Mishra, S., Zhang, H., Chen, X., Chen, M., Nova, A., Hou, L., Cheng, H.-T., Le, Q. V., Chi, E. H., and Zhou, D. NATURAL PLAN: Benchmarking LLMs on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024b.

Zheng, K., Han, J. M., and Polu, S. MiniF2F: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*, 2022.

Zhou, J. P., Staats, C. E., Li, W., Szegedy, C., Weinberger, K. Q., and Wu, Y. Don't trust: Verify–grounding LLM quantitative reasoning with autoformalization. In *International Conference on Learning Representations (ICLR)*, 2024a.

Zhou, J. P., Wu, Y., Li, Q., and Grosse, R. B. REFACTOR: Learning to extract theorems from proofs. In *International Conference on Learning Representations (ICLR)*, 2024b.