Trustless AI with the Lean Theorem Prover

Sebastian Ullrich
Head of Engineering, Lean FRO

July 11, 2025

How can we ensure confidence in statements made by humans or AI, and that every proof of correctness is independently verifiable?

Lean is a Development Environment for formal verification

Mathlib > RingTheory > Finiteness.lean

```
theorem FG.stabilizes_of_iSup_eq {M' : Submodule R M} (hM' : M'.FG) (N : ℕ →o Submodule R M)
    (H : iSup N = M') : ∃ n, M' = N n := by
  obtain ⟨S, hS⟩ := hM'
  have : ∀ s : S, ∃ n, (s : M) ∈ N n := fun s =>
    (Submodule.mem_iSup_of_chain N s).mp
      (by
        rw [H, ← hS]
        exact Submodule.subset_span s.2)
  choose f hf using this
  use S.attach.sup f
  apply le_antisymm
  · conv_lhs => rw [← hS]
    rw [Submodule.span_le]
    intro s hs
    exact N.2 (Finset.le_sup <| S.mem_attach ⟨s, hs⟩) (hf _)
  · rw [← H]
    exact le_iSup _ _
```

▼Finiteness.lean:365:2

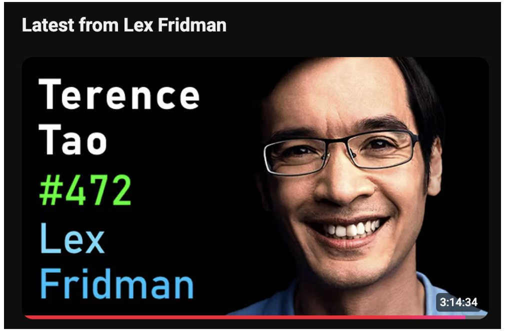
▼Tactic state

1 goal

```
▼case intro
R : Type u_1
M : Type u_2
inst✝² : Semiring R
inst✝¹ : AddCommMonoid M
inst✝ : Module R M
M' : Submodule R M
N : ℕ →o Submodule R M
H : iSup ⇑N = M'
S : Finset M
hS : span R ↑S = M'
f : { x // x ∈ S } → ℕ
hf : ∀ (s : { x // x ∈ S }), ↑s ∈ N (f s)
⊢ ∃ n, M' = N n
```

Lean is Taking Mathematics by Storm

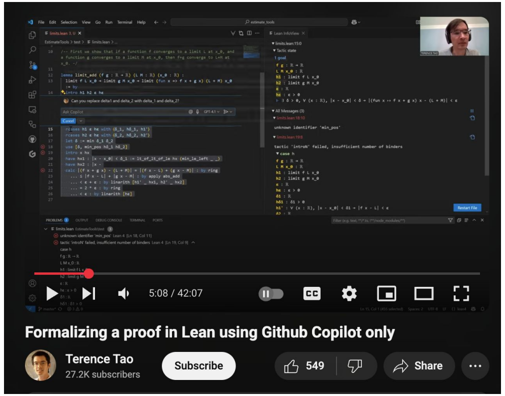
*"**Lean enables large-scale collaboration** by allowing mathematicians to break down complex proofs into smaller, verifiable components. This formalization process ensures the correctness of proofs and facilitates contributions from a broader community. **With Lean, we are beginning to see how AI can accelerate the formalization of mathematics, opening up new possibilities for research.**" — Terence Tao*



Fermat's Last Theorem – Kevin Buzzard

Carleson's Theorem – Floris van Doorn

…and many more!



AI Proof Assistants

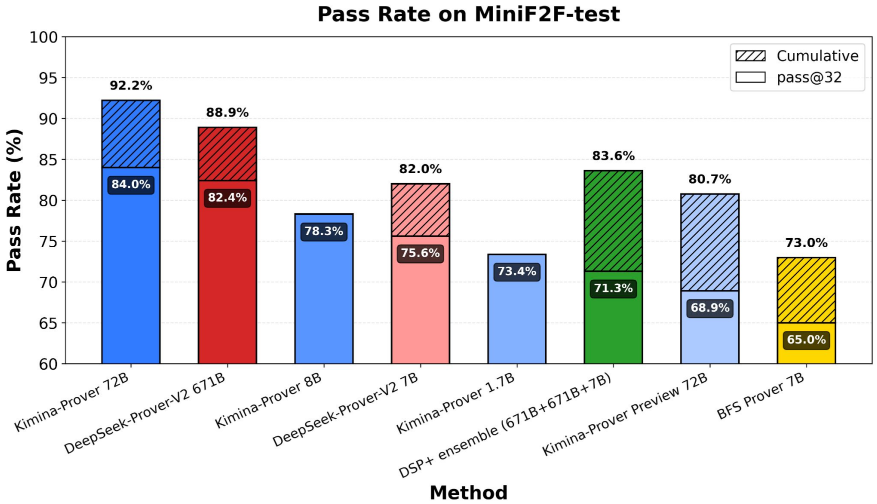
Several groups are developing AI that suggests the **next move**(s) in Lean's interactive proof game.

LeanDojo is an open-source project from Caltech, and everything (model, datasets, code) is open.

OpenAI and Meta AI have also developed AI assistants for Lean.

Claude 4 is fantastic on Lean code. Their System Card contains a Lean example.

DeepSeek-Prover-V2 and Kimina-Prover (released yesterday!) are designed for generating Lean proofs.



Lean Enables Verified AI for Mathematics and Code

LLMs are powerful tools, but they are prone to **hallucinations**.

In math, a **small mistake can invalidate the whole proof**.

Even if your AI system claims to have solved an open conjecture, who is going to verify it?

Machine-checkable proofs are the antidote to hallucinations.

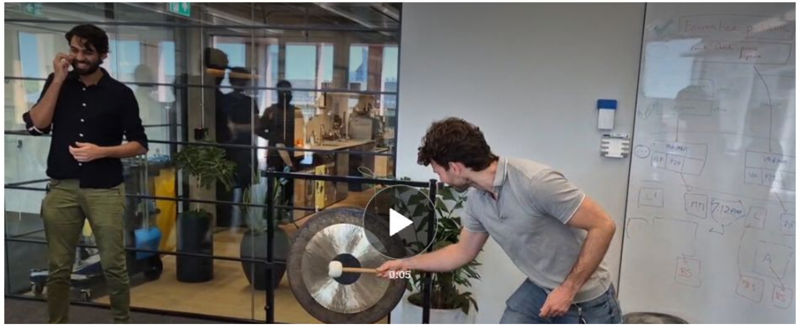
The New York Times

Artificial Intelligence › | A.I.'s Math Problem | A.I. Training Data Disappears | Microsoft's Risk-Taker | Fine Print Changes | Quiz: Fake or Real Images?

Move Over, Mathematicians, Here Comes AlphaProof

A.I. is getting good at math — and might soon make a worthy collaborator for humans.

Share full article | 47



"At Google DeepMind, we used Lean to build AlphaProof, a new reinforcement-learning based system for formal math reasoning. ***Lean's extensibility and verification capabilities were key in enabling the development of AlphaProof****."* — *Pushmeet Kohli, Vice President, Research Google DeepMind*

AlphaProof & the International Math Olympiad

Determine all real numbers α such that, for every positive integer n, the integer

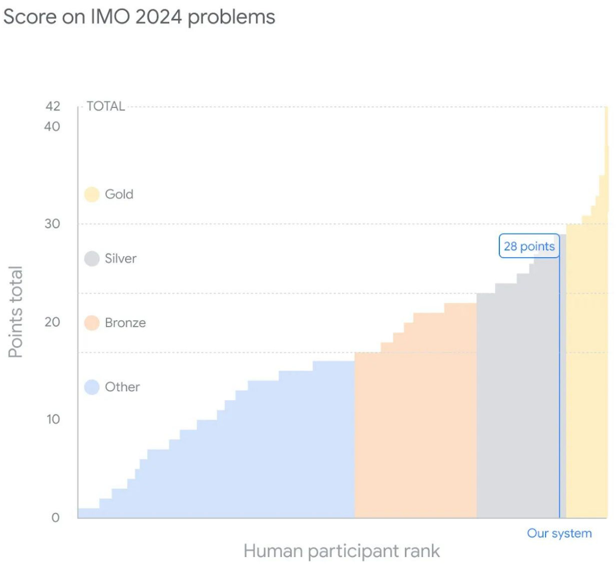
$$\lfloor \alpha \rfloor + \lfloor 2\alpha \rfloor + \cdots + \lfloor n\alpha \rfloor$$

is a multiple of n. (Note that $\lfloor z \rfloor$ denotes the greatest integer less than or equal to z. For example, $\lfloor -\pi \rfloor = -4$ and $\lfloor 2 \rfloor = \lfloor 2.9 \rfloor = 2$.)

Solution: α is an even integer.

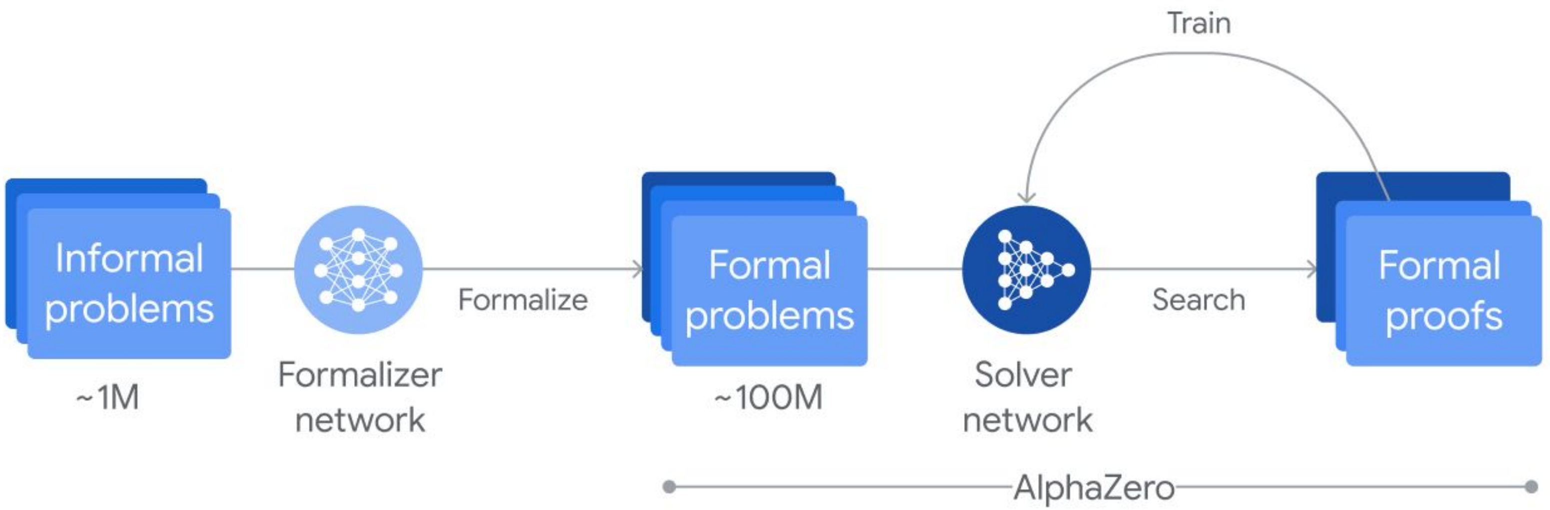
```
open scoped BigOperators

theorem imo_2024_p1 :
    {(α : ℝ) | ∀ (n : ℕ), 0 < n → (n : ℤ) | (∑ i in Finset.Icc 1 n, ⌊i * α⌋)}
    = {α : ℝ | ∃ k : ℤ, Even k ∧ α = k} := by
  rw [(Set.Subset.antisymm_iff ), (Set.subset_def), ]
  /- We introduce a variable that will be used
      in the second part of the proof (the hard direction)
```



deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level

Learning à la DeepMind



deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

Tencent AI & IMO

Zhenwen Liang • 2nd **+ Follow** ···
Research Scientist at Tencent AI Lab, Se...
10h •

🚀 TL;DR: 5 post-2000 IMO Problems Proved!

Excited to share our latest research at Tencent AI Lab, where we introduce a new framework for Automated Theorem Proving that tackles some of the world's most difficult math problems.

For the first time, we have successfully solved 5 post-2000 International Mathematical Olympiad (IMO) problems, a benchmark where no previous open-source prover had reported success.

Our key insight is to decouple high-level reasoning from formal proof generation. By using a powerful LLM as a "Reasoner" to strategize and a specialized model as a "Prover" to verify, we bridge the critical gap between informal intuition and formal rigor that has limited current AI systems.

To accelerate research in this area, we are also releasing a dataset of over 600 verified lemmas for challenging IMO problems. We believe this resource will provide a new foundation for the community to build upon.

tencent-imo.github.io

Autoformalization

README

The abc conjecture almost always — autoformalized

This is a completely machine-generated formalization of the classical theorem of de Bruijn, which bounds the exceptional set in the abc conjecture. We follow the proof laid out in this expository note.

All statements, proofs, and documentation were created by Trinity, an autoformalization system developed by Morph Labs as part of the Verified Superintelligence project.

github.com/morph-labs/lean-abc-true-almost-always

ImProver: Agent-Based Automated Proof Optimization

Original (Human-Written)

```
lemma lemma0 {α : Type} {p : α → α → Prop}
(h1 : ∀ x, ∃! y, p x y)
(h2 : ∀ x y, p x y ↔ p y x) :
∀ x, Classical.choose
    (h1 (Classical.choose (h1 x).exists)).exists=x :
  -- PROOF START
  intro x
  obtain ⟨y, h1e, h1u⟩ := h1 x
  have h2' : Classical.choose (h1 x).exists = y :=
    h1u _ (Classical.choose_spec (h1 x).exists)
  rw [h2']
  obtain ⟨w, h1e', h1u'⟩ := h1 y
  have h4 := Classical.choose_spec (h1 y).exists
  have hxw : x = w := by
    apply h1u'
    rw [h2]
    exact h1e
  rw [hxw]
  exact h1u' _ h4
```

ImProver (Length-Optimized)

```
lemma lemma0 {α : Type} {p : α → α → Prop}
  (h1 : ∀ x, ∃! y, p x y)
  (h2 : ∀ x y, p x y ↔ p y x) :
  ∀ x, Classical.choose
      (h1 (Classical.choose (h1 x).exists)).exists=x

  -- PROOF START
  intro x
  obtain ⟨y, h1e, h1u⟩ := h1 x
  rw [h1u _ (Classical.choose_spec _)]
  obtain ⟨w, h1e', h1u'⟩ := h1 y
  rw [h1u' _ ((h2 _ _).mpr h1e)]
  exact h1u' _ (Classical.choose_spec _)
```

riyazahuja.github.io/ImProver

Lean+AI preprints in May/June 2025

Prover Agent: An Agent-based Framework for Formal Mathematical Proofs**,** Baba et al

LeanTutor: A Formally-Verified AI Tutor for Mathematical Proofs, Patel et al

Safe: Enhancing Mathematical Reasoning in LLMs, Liu et al

VERINA: Benchmarking Verifiable Code Generation, Ye et al

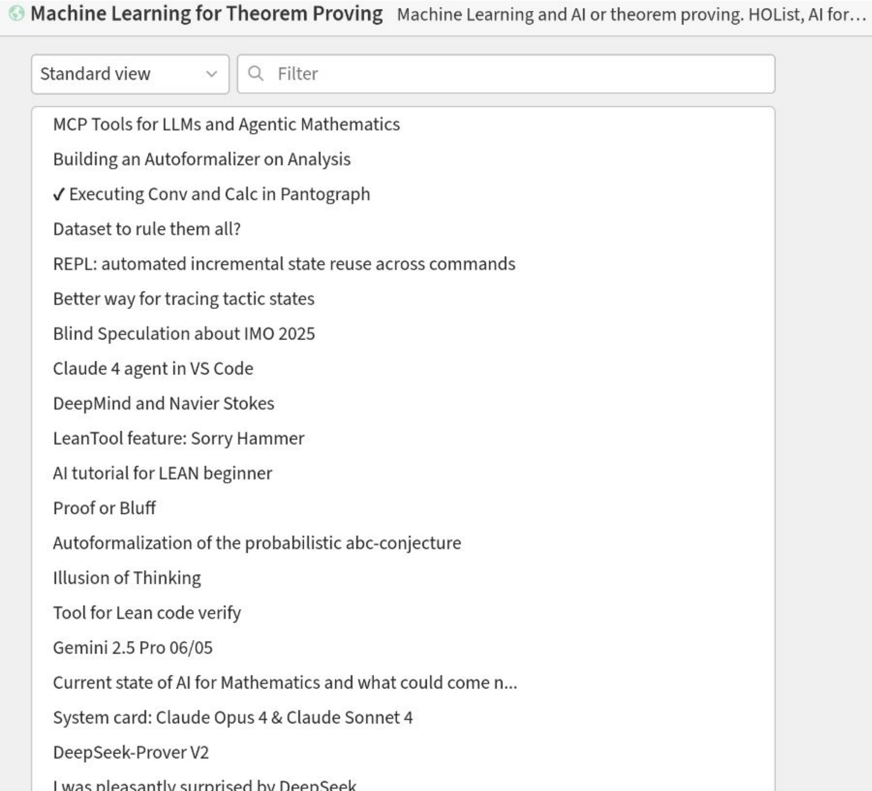
REAL-Prover: Retrieval Augmented Lean Prover for Mathematical Reasoning, Shen et al

Enumerate-Conjecture-Prove: Formally Solving Answer-Construction Problems in Math Competitions, Sun et al

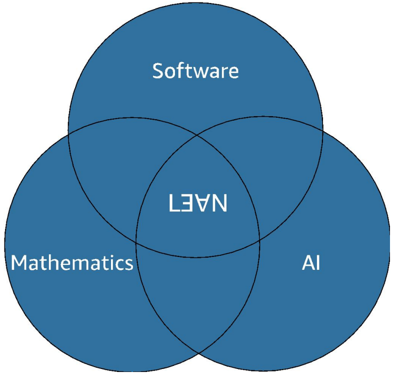
APOLLO: Automated LLM and Lean Collaboration for Advanced Formal Reasoning, Ospanov et al

FormalMATH: Benchmarking Formal Mathematical Reasoning of Large Language Models, Yu et al

A vibrant community of users at leanprover.zulipchat.com



Conclusion



Lean is a development environment for formalized mathematics and program verification

Lean's rigor allows for *trustless* collaboration between humans and AI

- AI for accelerating Lean development
- Lean for verifying AI output

Lean users are benefiting from AI today, and more research arrives weekly

Thank You

https://leanprover.zulipchat.com/
x: @leanprover
LinkedIn: Lean FRO
Mastodon: @leanprover@functional.cafe
#leanlang, #leanprover

https://www.lean-lang.org/