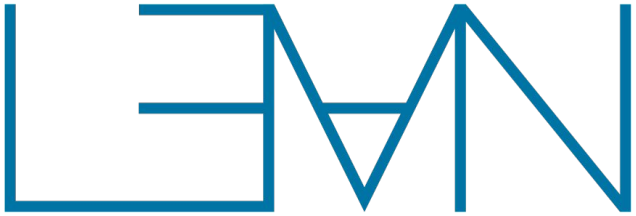
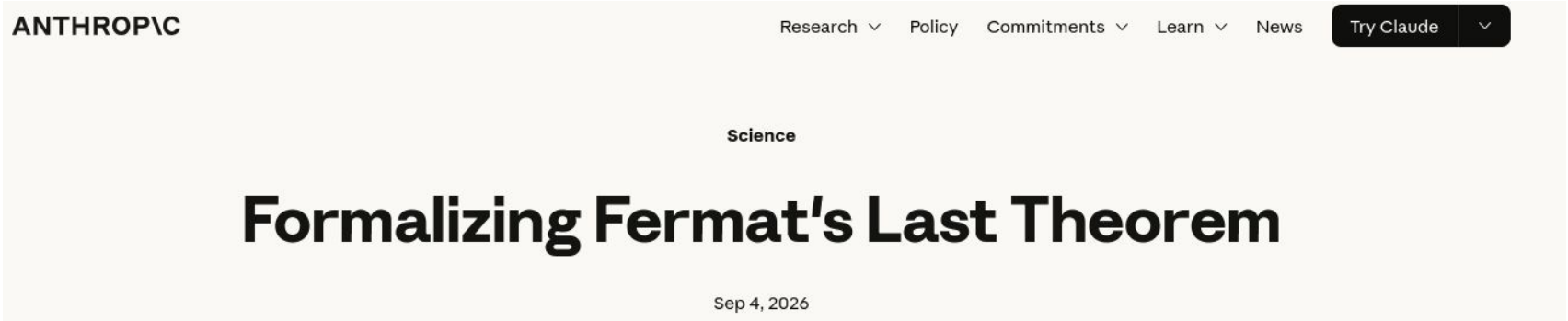


Lean and trust in the age of AI

Sebastian Ullrich
Head of Engineering, Lean FRO

IFIP WG 2.2 Meeting, Brussels, Sep 17, 2026

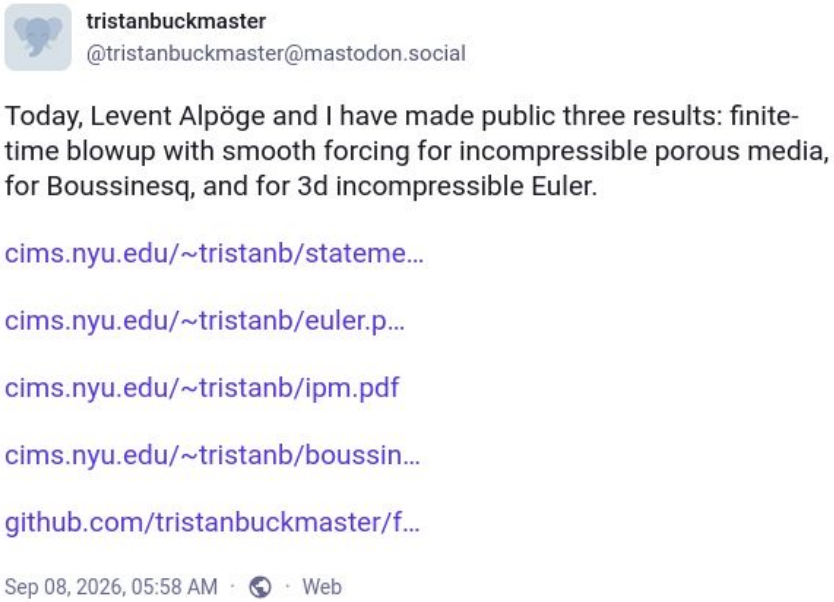
Sep 4 …



*"We are sharing the first complete computer-checked proof of Fermat's Last Theorem. Claude worked largely autonomously over 11 days to write the proof in the **Lean programming language**."*

anthropic.com/research/formalizing-fermats-last-theorem

… Sep 7 …



tristanbuckmaster
@tristanbuckmaster@mastodon.social

Today, Levent Alpöge and I have made public three results: finite-time blowup with smooth forcing for incompressible porous media, for Boussinesq, and for 3d incompressible Euler.

cims.nyu.edu/~tristanb/stateme...

cims.nyu.edu/~tristanb/euler.p...

cims.nyu.edu/~tristanb/ipm.pdf

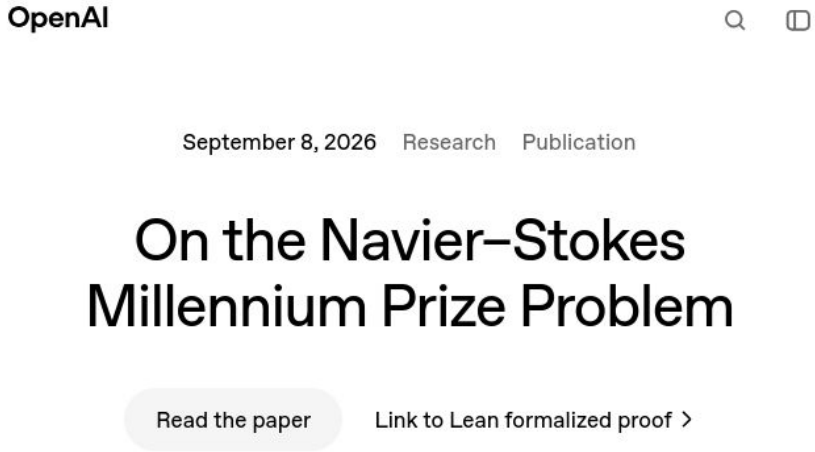
cims.nyu.edu/~tristanb/boussin...

github.com/tristanbuckmaster/f...

Sep 08, 2026, 05:58 AM · · Web

mastodon.social/@tristanbuckmaster/117233413705701198

… Sep 8 …



OpenAI

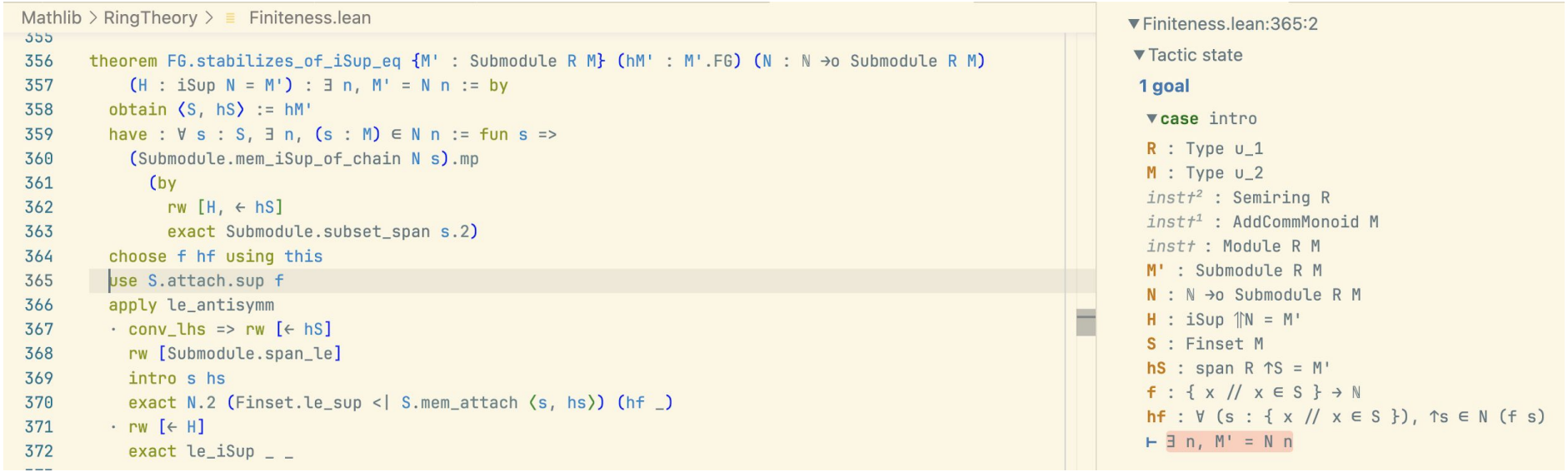
September 8, 2026 Research Publication

On the Navier–Stokes Millennium Prize Problem

Read the paper Link to Lean formalized proof >

openai.com/index/navier-stokes-solution

Lean is a Development Environment for Formal Verification



How did we get here?

The IMO Grand Challenge (2023)

IMO Grand Challenge

The International Mathematical Olympiad (IMO) is perhaps the most celebrated mental competition in the world and as such is among the ultimate grand challenges for Artificial Intelligence (AI).

The challenge: build an AI that can win a gold medal in the competition.

Committee

- Daniel Selsam (OpenAI)
- Leonardo de Moura (Microsoft Research)
- Kevin Buzzard (Imperial College London)
- Reid Barton (University of Pittsburgh)
- Percy Liang (Stanford University)
- Sarah Loos (Google AI)
- Freek Wiedijk (University of Nijmegen)

AlphaProof & the International Math Olympiad (2024)

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

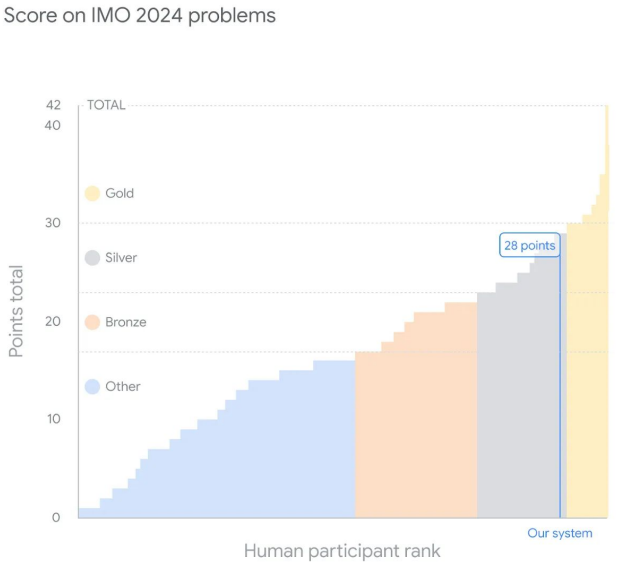
$$\lfloor \alpha \rfloor + \lfloor 2\alpha \rfloor + \cdots + \lfloor n\alpha \rfloor$$

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

Article | Published: 12 November 2025

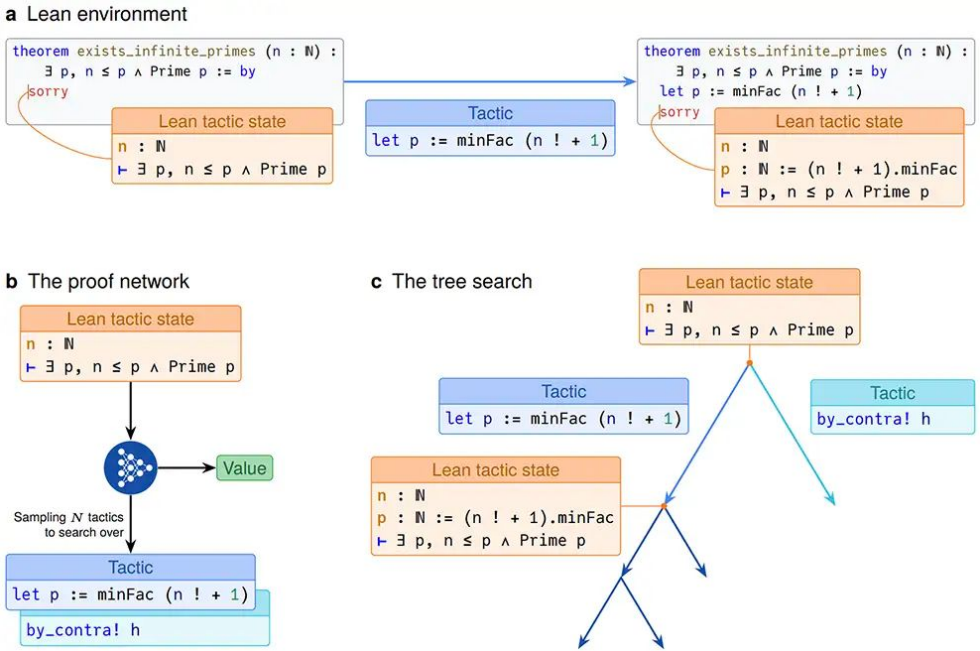
Olympiad-level formal mathematical reasoning with reinforcement learning

Thomas Hubert ✉, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser ✉, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli & David Silver — Show fewer authors

Nature (2025) | Cite this article

nature.com/articles/s41586-025-09833-y

Learning à la DeepMind



julian.ac/blog/2025/11/13/alphaproof-paper

IMO 2025: 3 Gold, 1 Silver

OpenAI (informal)

DeepMind (informal)

Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad

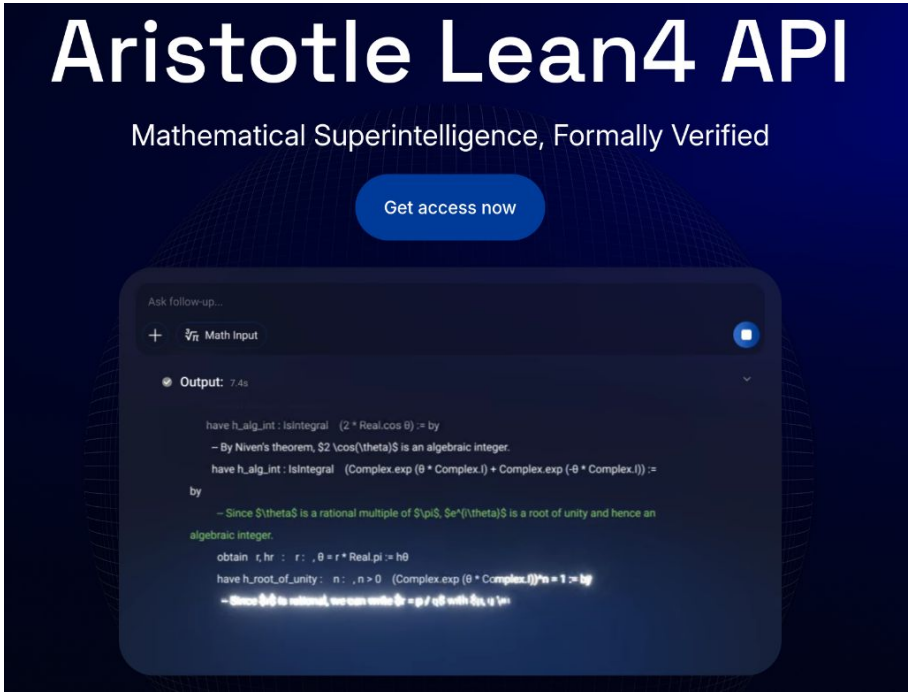
Harmonic (Lean)

Aristotle Achieves Gold Medal-Level Performance at the International Mathematical Olympiad, iOS App Beta Launch

ByteDance (Lean)

ByteDance Seed Prover Achieves Silver Medal Score in IMO 2025

IMO-Level Proof Search as a Service



AlphaProof Interest Form

Thanks for your interest in AlphaProof. To deliver a reliable user experience, we are carefully managing our compute resources to maintain system stability. We will be reviewing applications and prioritizing access over time as we scale the system. We appreciate your patience and understanding.

To help you get the most from this process, we encourage you to first explore the AlphaProof Blog Post, the AlphaProof Nature Paper, and the **AlphaProof Tool Demos [1][2]**. Understanding AlphaProof beforehand will empower you to complete this form more effectively and confidently.

aristotle.harmonic.fun

deepmind.google/alphaproof

2026: Putnam Bench Saturated

★ FEATURED RANKING

Lean — all 672 solved (cheapest first)

#	Model	num-solved	compute	date added
1	NEAR AI (w/ DeepSeek V4)💚	672	Mean $0.17, median $0.04, max $11.18 per problem	2026-09-04
2	Humanfia💙	672	Avg $44.5 total cost/ problem	2026-08-27
3	Aleph Prover (Logical Intelligence)	672	Avg $74, Max $1468 per problem	2026-08-26

💚 fully open-sourced

💙 partially open-sourced

trishullab.github.io/PutnamBench/leaderboard.html

Extensibility and Introspection

*"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*

Lean 4 is implemented in Lean, allowing for unprecedented extension and introspection by users.

leanprover-community/repl is a simple Lean program for communicating with the Lean compiler via JSON.

AI labs have customized it in many ways, especially around parallelized proof tree search.

stanford-centaur/PyPantograph is the state of the art for open-source Lean REPLs.

Autoformalization (2025)

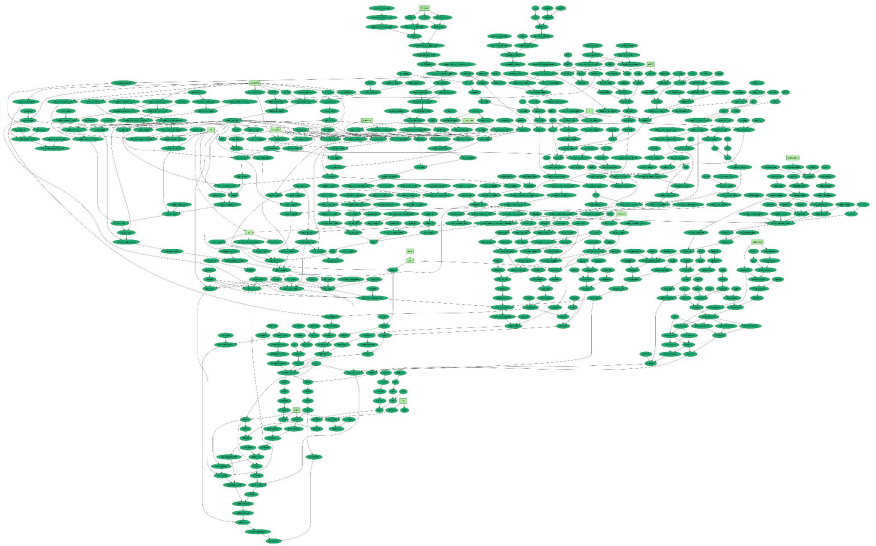
Auguste Poiroux

Dear all,

We, at Math, Inc., are excited to introduce Gauss, a new autoformalization agent designed to assist mathematicians in their Lean formalization.

Using Gauss, and building over the recent medium PNT progress by Kontorovich et al., we formalized the **strong Prime Number Theorem** in Lean. Over the course of three weeks, Gauss produced ~25k lines of Lean code and over 1,000 theorems and definitions. To put these numbers in perspective, our previous iteration, three months ago, produced ~3.5k lines of Lean code and 100 theorems and definitions to formalize the abc conjecture almost always.

Our goal is to make Gauss accessible and easy to use. If you are interested, you can register for early access to Gauss here. We are happy to discuss about Gauss and Math, Inc. in the discussion thread.



github.com/math-inc/strongpnt

Vibe Proving (2025)

Forbidden Sidon subsets of perfect difference sets, featuring a human-assisted proof

Boris Alexeev ChatGPT* Lean† Dustin G. Mixon‡§

Abstract

We resolve a $1000 Erdős prize problem, complete with formal verification generated by a large language model.

In over a dozen papers, beginning in 1976 and spanning two decades, Paul Erdős repeatedly posed one of his "favourite" conjectures: every finite Sidon set can be extended to a finite perfect difference set. We establish that $\{1, 2, 4, 8, 13\}$ is a counterexample to this conjecture.

During the preparation of this paper, we discovered that although this problem was presumed to be open for half a century, Marshall Hall, Jr. published a different counterexample three decades *before* Erdős first posed the problem. With a healthy skepticism of this apparent oversight, and out of an abundance of caution, we used ChatGPT to vibe code a Lean proof of both Hall's and our counterexamples.

borisalexeev.com/pdf/erdos707.pdf

Not Just Mathematics

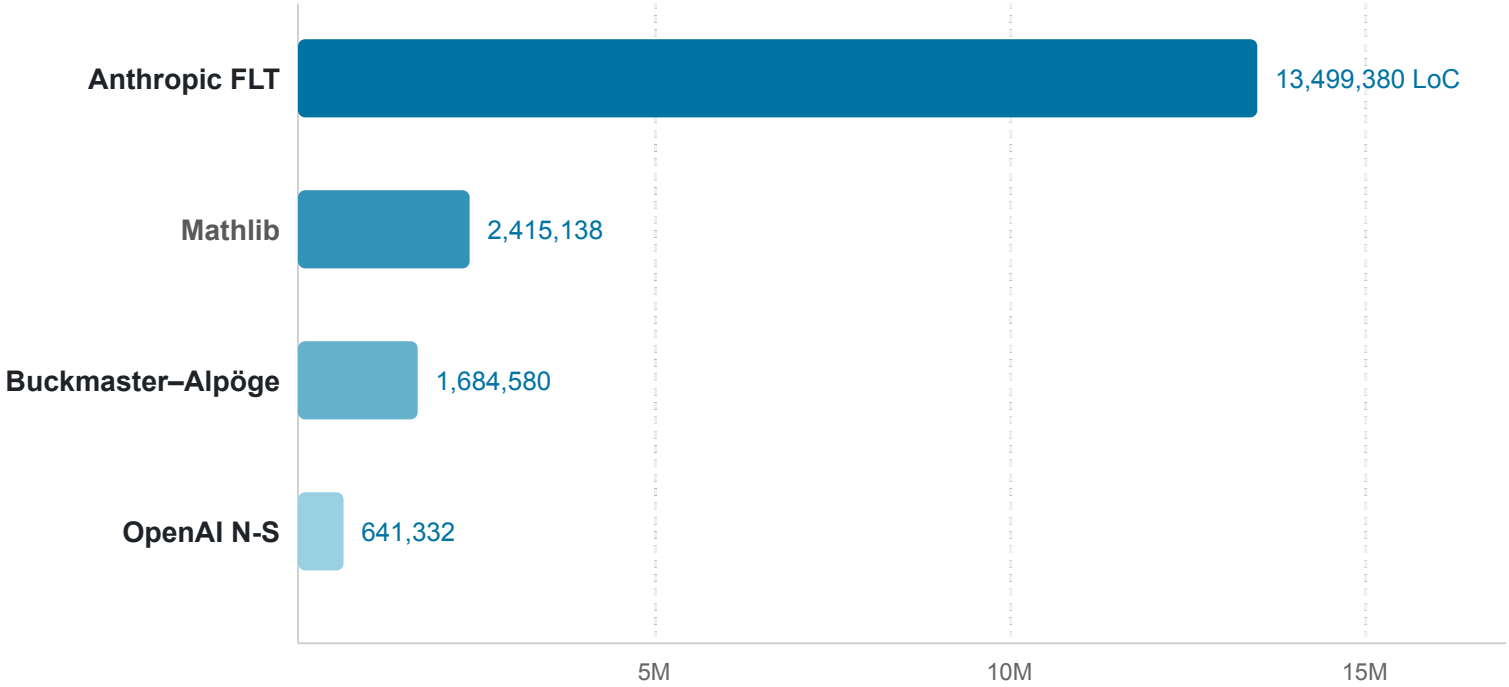
SymCrypt is Microsoft's core cryptographic library, being rewritten in Rust and verified with Lean and AI.

"We are verifying code faster than we can write it." — Son Ho, Azure Research

lean-zip (Kim Morrison, Lean FRO) is a formally verified and optimized DEFLATE implementation in Lean, implemented entirely by loosely supervised AI.

Where do we go from here?

That Question of Trust



Verification

Check it yourself

- You need Linux or macOS (some paths are too long for Windows), elan (it installs Lean 4.33.1 from `lean-toolchain`), and a network connection: Lake fetches Mathlib from GitHub and compiles it from source, since no prebuilt Mathlib matches this toolchain (about 13 minutes at 96 jobs).
- The build needs about 5 GB of memory per parallel job (a few modules need up to 36 GB); about 67 GB of disk under `.lake/`, plus C files (about 220 GB) that can be deleted as the build goes. Ours took 5 h 32 min at 96 jobs, with a peak of 153 GB of memory.
- comparator takes about 15 hours (ours: 14 h 46 min), nearly all of it the kernel replay on one core. Our peak memory was 230 GB, so allow 300 GB. Run nanoda after the comparator script, whose tools it reuses. Writing the 37.8 GB export takes about 90 GB of memory for an hour, and the check itself about 40 GB (about 30 minutes at 16 threads). Both scripts are for Linux (bash, git, python3, GNU coreutils; nanoda also needs `patch`, `cargo` and crates.io).

github.com/anthropics/fermats-last-theorem#check-it-yourself

Verification, faster

state	commit	modules	instructions	task-clock	wall	sys	max RSS, one process	peak anon, whole build
baseline, lakefile only	3c366b27	1,000	39,952 G	8,866 s	1,022 s	~2,902 s	~8,075 MiB	~21.5 GiB
small cone transformed, local fixes, repairs	16ccdde0	960	31,613 G (−20.9%)	7,759 s (−12.5%)	907 s (−11.2%)	~2,916 s	~7,740 MiB	~20.3 GiB
olean-export fix	0fda8bdc	960	21,047 G (−47.3%)	7,128 s (−19.6%)	287 s (−71.9%)	~2,951 s	~7,636 MiB	~21.1 GiB
Modulize, private imports, merge	7197a933	540	17,029 G (−57.4%)	4,542 s (−48.8%)	209 s (−79.5%)	~1,216 s	~4,335 MiB	~16.4 GiB
shake	ae6a3ebf	540	14,981 G (−62.5%)	3,487 s (−60.7%)	171 s (−83.3%)	~603 s	~3,166 MiB	~17.3 GiB
rev-use cleanup	**fb23f1af**	**540**	**13,643 G (−65.8%)**	**3,268 s (−63.1%)**	**140 s (−86.3%)**	**~606 s**	**~2,859 MiB**	**~13.7 GiB**

leanprover.zulipchat.com/#narrow/channel/416277-FLT/topic/Anthropic.20formalization.3A.20code.20analyses

Recapping the Trusted Code Base

In theory, you only have to trust the statement of a proof and the kernel.

In practice, the meaning of statements can be affected by preceding/imported code… and all executed code may potentially influence everything else anyway.

Isolation, in multiple dimensions, of statement and proof is a necessity with untrusted inputs.

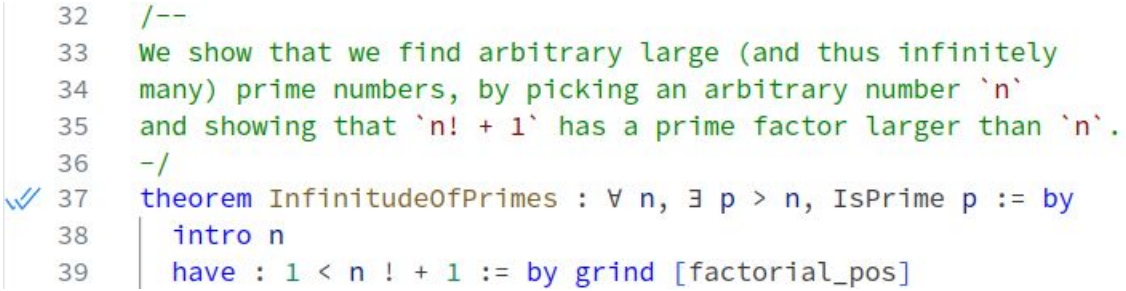
Validating a Lean Proof (Lean Reference Manual)

Breaks the trust question down into four incremental checking procedures

lean-lang.org/doc/reference/latest/ValidatingProofs

Validating a Lean Proof: Level I

The "blue double check marks" signify in-process acceptance by the official C++ kernel



```
/--
We show that we find arbitrary large (and thus infinitely
many) prime numbers, by picking an arbitrary number `n`
and showing that `n! + 1` has a prime factor larger than `n`.
-/
theorem InfinitudeOfPrimes : ∀ n, ∃ p > n, IsPrime p := by
  intro n
  have : 1 < n ! + 1 := by grind [factorial_pos]
```

Protects against "innocent" mistakes:

- incomplete proofs/`sorry`
- tactics producing incorrect proof terms

… **in the current theorem**

Validating a Lean Proof: Level II

`#print axioms` lists all axioms (transitively) used by a theorem.

Three built-in axioms relied on pervasively in Lean: choice, propositional extensionality, quotient lifting

The Gold Standard of the pre-AI world.

Protects against

- **inherited** incomplete/incorrect proof terms
- custom axioms

Validating a Lean Proof: Level III

The kernel can be run as its own process: `leanchecker`

Protects against

- incorrect kernel handling by Lean (e.g. imported declarations are trusted)
- metaprograms bypassing the in-process kernel or otherwise corrupting processing

Validating a Lean Proof: Level IV

`comparator` **isolates** statement processing, proof processing, and `leanchecker` into separate, **sandboxed** processes. Optionally runs additional checkers as well.

Protects against

- actively malicious proofs (but not incorrect statements)
- bugs in some but not all checkers

Validating a Lean Proof: Level IV

A **Challenge.lean** file contains only the statement with minimal dependencies for review.

```
theorem FLT_for_comparator (n : ℕ) (hn : 3 ≤ n) (a b c : ℕ) (ha : 0 < a) (hb : 0 < b) (hc : 0 < c) :
    a ^ n + b ^ n ≠ c ^ n :=
  sorry
```

Solution.lean then is checked to be a refinement of the challenge with arbitrary further imports.

The solution is accepted if it passes all checkers and axiom checks (i.e. no `sorry`) and its statement plus relevant closure is equal to that in the challenge.

Sourcing Statements

FLT can be stated without additional dependencies. Mathlib provides the statement(!) for any semiring.

google-deepmind/formal-conjectures provides the statement of **Navier-Stokes** used by OpenAI.

Buckmaster-Alpöge provide three challenge files importing only Mathlib.

External Checkers

The **Lean Kernel Arena** collects, checks, and profiles external checkers.

At the time of writing, the #1 spot **sokonanoda** (Jeremy Chen, Cambridge) checks Mathlib **17x** as fast as the official kernel.

lean4lean (Mario Carneiro, Chalmers) is a notable port and in-progress formalization of the official kernel and the type theory in Lean itself. Some overlap and cooperation with the **MetaRocq** (INRIA) project.

con-leche (Joachim Breitner, Lean FRO) is a kernel AI-written in Lean with verified consistency, by way of a set-theoretic model. **con-ron** is a verified port to Rust, removing the Lean runtime from the TCB.

comparator, lean4lean, con-leche, and con-ron come **bundled with Lean 4.35+**.

Revisiting the Trusted Code Base at Level IV

- the statement (small and independent, ideally)
  - plus its elaboration
- at least one of the used checkers (impl ~7,000 LoC; verification statement ~100 LoC)
- comparator (~1,000 LoC)
  - plus its compilation and runtime
- the OS sandbox (`bubblewrap`), the OS, the hardware (*many* LoC)

The newest version of comparator allows for producing the proof export file manually, such as in a VM or on a separate machine.

Lean FRO: Shaping the Future of Lean Development

The Lean Focused Research Organization (FRO) is a non-profit dedicated to Lean's development.

Founded in **August 2023**, the organization has 19 members.

Its mission is to enhance critical areas: **scalability**, **usability**, **documentation**, and **proof automation**.

It must reach **self-sustainability by August 2028** and become the **Lean Foundation**.

We are very grateful for all philanthropic support we have received.

Conclusion

The advent of AI proofs has completely changed the trust question for theorem provers.

The default assumption now is that if an agent *could* feasibly find a weakness, it will use it.

Lean is the first major ITP to ship with an isolated checking setup and verified kernel.

Thank You

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

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