Formalized mathematics in the age of AI

Floris van Doorn

University of Bonn

11 Nov 2026







Outline

- Formalization
- AI in formalization
- Trust in proofs

Outline

- Formalization
- AI in formalization
- Trust in proofs

Disclaimers:

- Opinions on AI differ
- With AI I mean "generative AI" (like LLMs)
- I don't have all the answers

Some data analysis was performed and LaTeX help was given by generative AI in the preparation of these slides

Formalization of mathematics

- Digitization of mathematical definitions, theorems and proofs.
- A proof assistant is a program that can check the correctness of proofs.
- Examples include Lean, Rocq, Isabelle, ...

Lean

Lean's development started in 2013 by Leonardo de Moura.

Currently maintained and developed by the Lean FRO with ~ 20 developers.



Implements a flexible language for programming and formalized proof.

Example

```
example {X Y Z : Type*} [TopologicalSpace X] [TopologicalSpace Y] [TopologicalSpace Z]
    {g : Y → Z} {f : X → Y} (hg : Continuous g) (hf : Continuous f) : Continuous (g ∘ f) := by
  -- Let us show that `g ∘ f` is continuous
  constructor
  -- We have to show that if `U ⊆ Z` is open, then the preimage `(g ∘ f)⁻¹(U)` is open
  show ∀ U : Set Z, IsOpen U → IsOpen ((g ∘ f) ⁻¹' U)
  -- Suppose `U ⊆ Z` is open
  intro (U : Set Z) (hU : IsOpen U)
  -- By continuity of `g`, `g⁻¹(U)` is open
  have : IsOpen (g ⁻¹' U) := hU.preimage hg
  -- By continuity of `f`, `f⁻¹(g⁻¹(U))` is open
  have : IsOpen (f ⁻¹' g ⁻¹' U) := this.preimage hf
  -- `f⁻¹(g⁻¹(U)) = (g ∘ f)⁻¹(U)`
  have : f ⁻¹' g ⁻¹' U = (g ∘ f) ⁻¹' U := rfl
  -- This finishes the proof
  grind
```

Lean's mathematical library

Mathlib is Lean's mathematical library.

It is monolithic, containing algebra, analysis, geometry, number theory, probability theory, combinatorics, logic, topology, category theory, ...

It is large: Mathlib has 2.5 million lines of code, with 290k "theorems" and 140k "definitions" written by over 750 contributors.

It is actively developed: There are about 200 contributions every week, reviewed by the maintainers and reviewers.

I have worked with Lean since 2014 and am a maintainer of Mathlib since 2019.

Some human-led formalization projects

- Kepler conjecture (Hales et al.; 2014; HOL Light / Isabelle)
- Independence of the Continuum Hypothesis (vD, Jesse Han; 2019)
- Perfectoid Spaces (Buzzard, Commelin, Massot; 2020)
- Liquid Tensor Experiment (led by Commelin and Topaz; 2022)
- Polynomial Freiman–Rusza conjecture (led by Terence Tao; 2023)
- Sphere Eversion (vD, Massot, Nash; 2023)
- Prime Number Theorem+ (led by Kontorovich and Tao; 2025)
- Equational theories (led by Tao, 34 authors; 2025)
- Fermat's Last Theorem (led by Buzzard; Ongoing)
- Carleson's Theorem (led by vD; 2025)
- Sphere Packing (Birkbeck, Hariharan, Lee, Ma, Mehta, Viazovska; 2026/ongoing)

AI + Lean

Generative AI systems can increasingly write Lean formalizations

- Jul 2024: 3 IMO problems
- Jun 2025: De Bruijn's theorem on the abc-conjecture
- Mar 2026: Sphere packing in 24 dimensions
- May 2026: Erdős' unit-distance conjecture is false
- Sep 2026: Fermat's Last Theorem
- Sep 2026: Blowup in the Navier–Stokes equations
- Oct 2026: OpenAI's 722 manuscripts, ~300 formalized
- Oct 2026: Classification of finite simple groups
- Nov 2026: Stacks project

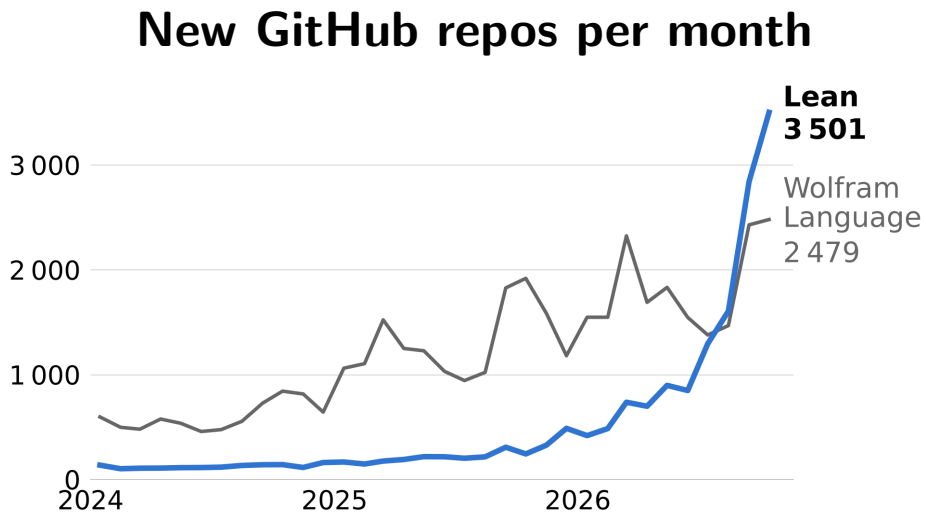
AI + Lean

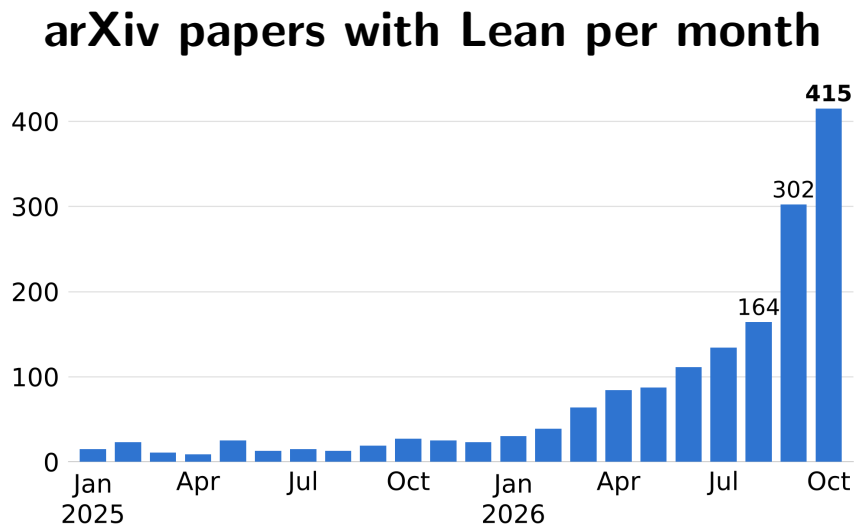
Generative AI systems can increasingly write Lean formalizations

- Jul 2024: 3 IMO problems (170 LoC)
- Jun 2025: De Bruijn's theorem on the abc-conjecture (3500 LoC)
- Mar 2026: Sphere packing in 24 dimensions (200k LoC)
- May 2026: Erdős' unit-distance conjecture is false (1.2M LoC)
- Sep 2026: Fermat's Last Theorem (13M LoC)
- Sep 2026: Blowup in the Navier–Stokes equations (340k LoC)
- Oct 2026: OpenAI's 722 manuscripts, ~300 formalized (26M LoC)
- Oct 2026: Classification of finite simple groups (43M LoC)
- Nov 2026: Stacks project (16M LoC)

(LoC is a bad measure; but the order of magnitude is indicative)

Lean usage is skyrocketing





Sources: GitHub search; `paperswithlean.com` (retrieved 10 Nov 2026)

- arXiv papers using Lean: ~ 90 in 2015–2024, ~ 220 in 2025, $400+$ in October 2026 alone
- Most recent papers have an AI-generated formalizations
- It is now relatively easy to use AI to generate a Lean-formalization of a research paper.
- Of course, generative AI can now also solve a lot of mathematical problems.

AI in formalization projects

- Mathlib/CSLib: big core libraries
  - contributions can use AI, but undergo a strict manual review; AI somewhat discouraged
- Formal Conjectures: collects statements of conjectures
  - lots of AI-generated contributions; manual review
- Tau Ceti: larger AI library on top of Mathlib
  - human-reviewed roadmaps, the rest is AI: writing, reviewing, refactoring code
- Most AI-formalization projects are one-offs and see little reuse.

Palomar is a registry of (mostly AI-generated) formal proofs.

Quantity versus quality

AI-generated formalizations

- \+ fast and cheap
- \+ correct: checked by Lean
- – long and hard to read
- – ad-hoc definitions, duplication
- – brittle

Canonical Mathlib formalizations

- \+ general, reusable definitions
- \+ coherent library
- \+ maintainable
- – time-intensive to write
- – human review is the bottleneck

Quantity versus quality

AI-generated formalizations

- \+ fast and cheap
- \+ correct: checked by Lean
- – long and hard to read
- – ad-hoc definitions, duplication
- – brittle
- \+ AI is getting better each month

Canonical Mathlib formalizations

- \+ general, reusable definitions
- \+ coherent library
- \+ maintainable
- – time-intensive to write
- – human review is the bottleneck

Quantity versus quality

Kim Morrison and Remy Degenne (Oct 28): *Simple quality indicators for Lean libraries*

- Uses AI to check for unnecessary hypotheses and duplication of arguments in proofs.
- Rough takeaway: Mathlib scores best; Tau Ceti reasonably well; other AI-generated libraries quite poorly.
- Result is naive and imperfect, but likely contains a signal

Why formalize?

- Verify correctness of results
  - AI can do this now much quicker
  - Enables us to trust new AI-generated mathematical results
  - OpenAI retracted 3 preprints with a critical flaw
- Write a comprehensive digital mathematical library
  - AI cannot reach the level of Mathlib yet
- Ease the burden of reviewing
  - ... though publishing will have to be rethought in the age of AI.
- Enable large-scale collaborations

Can we trust a formalized proof?

A formalization does not give absolute trust:

- Are the formalized statements correct? Are the definitions correct?
- Does Lean compile the proof?
- Is there any "cheating attempt"?

Is the statement correct?

For single theorems, this can *often* be established quickly.
However:

- statements can be complicated: CFSG
- there can be many statements: formalized Stacks project

Is the statement correct?

For single theorems, this can *often* be established quickly.
However:

- statements can be complicated: CFSG
- there can be many statements: formalized Stacks project

Lean statements can sometimes be misleading

- Lean/Mathlib uses "junk values"
  - $1/0$ is defined to be 0
  - $\int_{\mathbb{R}} f(x)\,dx = 0$ if f is not integrable
- There can be multiple versions of definitions
  - Manifolds in Lean can have boundary and are not required to be Hausdorff or finite dimensional
- It is easy to forget hypotheses or miss edge cases

Example: Navier–Stokes

OpenAI used the Lean statement from Formal Conjectures for the formalization of Navier–Stokes.

So can we conclude that the proof in the preprint is correct?

Example: Navier–Stokes

OpenAI used the Lean statement from Formal Conjectures for the formalization of Navier–Stokes.

So can we conclude that the proof in the preprint is correct?

Intermediate lemma statements diverged in the preprint compared to the formalization.

- *Navier–Stokes lost in translation* (Bastounis, Circelli, Hansen; 6 Oct 2026)

Does Lean compile the proof?

What happens when running Lean?

- Parsing and macro expansion (interpreting notation)
- Elaboration
  - Infer implicit information (unification, type-class inference, coercion insertion, overload resolution, ...)
  - Tactic execution

  This generates a "proof term".
- Kernel checking (small trusted core checker)

Catch cheating attempts

There are ways to "cheat" with Lean:

- You can declare new axioms
- You can use metaprogramming to circumvent the kernel
- You can use misleading notation/redefine the meaning of notation
- You can use Lean's programming capabilities to change files on your disk (e.g. the Lean executable)

Catch cheating attempts

Gold standard: `comparator` (comes with Lean)

- You have to write (or carefully review) a trusted "Challenge" file with the statement of the theorem(s)
- You have to provide a proof (can be large, untrusted, AI-generated)

Running `comparator`

- builds proof in sandboxed environment
- checks that the proof is complete and doesn't introduce new axioms.
- checks that the final statement matches the "Challenge" file
- exports the proof term
- uses $4+$ external type-checkers to check the proof

`comparator` protects against all known ways to cheat the system.

Bugs in the kernel

- The Lean kernel likely has bugs (7+ found by AI this summer)
- AI is very good at finding bugs
- multiple independent type-checkers

Bugs in the kernel

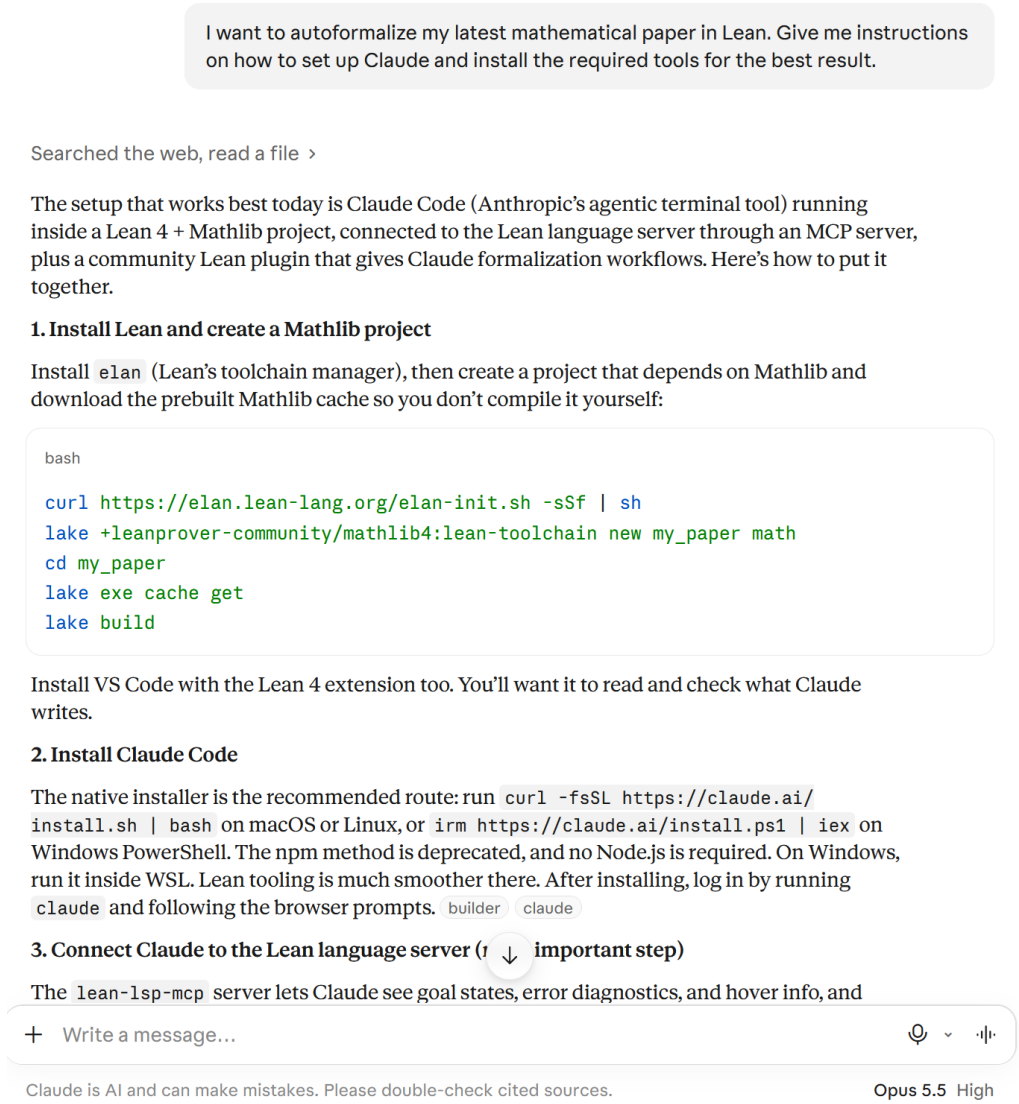
- The Lean kernel likely has bugs (7+ found by AI this summer)
- AI is very good at finding bugs
- multiple independent type-checkers

`con-leche` is an (AI-written) independent kernel with a formalized correctness proof.

- Yes, this is a circular argument
- Assumes a model of set theory with ω inaccessible cardinals.
- There can still be bugs in other security-critical parts, e.g. the Lean compiler.

Recommendation: use Lean

Easy: ask an LLM to formalize your recent preprint or paper.



Learning Lean

Some mathematicians should learn Lean:

- to verify that formalized statements is correctly formalized
- to make principled design decisions for formalized code
- to further develop and improve Mathlib, as a basis for other formalization projects

Students should learn some Lean, like LaTeX.

Future outlook

My opinions/beliefs:

- AI will continue to improve in proving mathematical results and formalizing them.
- AI capabilities are "jagged", and I expect this will remain the case, at least for a while.
- AI costs will come down; but there is a real risk of centralized power
- We will have to rethink many parts of mathematical practice: publishing, hiring, attribution, teaching, ...
- Human understanding of mathematics remains important

Conclusion

- AI is transforming mathematical practice
- Lean helps to identify correct proofs
- It is only part of the broader question how to rethink our mathematical practices

Thank you for listening

`https://leanprover-community.github.io/`

Jaggedness of AI code (source code for this presentation):

```
\includegraphics[height=0.47\linewidth,trim={0 30 0 69},clip]
  {figures/lean_vs_wolfram_github.png}
```

Jaggedness of AI code (source code for this presentation):

```
\includegraphics[height=0.47\linewidth,trim={0 30 0 69},clip]
  {figures/lean_vs_wolfram_github.png}
```

I asked it to trim and resize an imagine;
before this change, the code was

```
\includegraphics[width=\textwidth]
  {figures/lean_vs_wolfram_github.png}
```