

The role of mathlib in the age of AI

Michael B. Rothgang (he/him)

Formalised mathematics group, Universität Bonn



Formalised mathematics seminar, Uni Bonn
October 23, 2026

Outline of today's talk

- 1 Lean and mathematical formalization
- 2 Mathlib and the formalization ecosystem

Section 1

Lean and mathematical formalization

What is Lean?

Lean is

- a functional programming language
- a theorem prover

What is Lean?

Lean is

- a functional programming language
- a theorem prover

In Lean you can

- write programs/definitions
- state theorems
- write proofs
- write programs that write proofs $\rightarrow$ tactics

Program verification, formal mathematics

What is Lean?

Is it $\text{\LaTeX}$?

- No: $\text{\LaTeX}$ typesets mathematics, but does not check correctness

Is it like Sage or Mathematica?

- No: those programs perform computations, you don't build proofs in them

What is Lean?

Is it $\text{\LaTeX}$?

- No: $\text{\LaTeX}$ typesets mathematics, but does not check correctness

Is it like Sage or Mathematica?

- No: those programs perform computations, you don't build proofs in them

Is it AI that proves theorems automatically?

- No, but such AI can be built on top of proof assistants

Is it a program that automatically proves theorems in first order logic?

- No, but it can use such programs in tactics

Lean is not AI

You can use AI to write Lean code.

Using AI is neither sufficient nor necessary to write Lean.

“Vibe-writing” code without any understanding is not recommended.

Formalization and correctness

Every field has folklore results (no written proof), or literature gaps known only to experts.

By and large, the foundations of mathematics are sound.

Formalization and correctness

Every field has folklore results (no written proof), or literature gaps known only to experts.

By and large, the foundations of mathematics are sound.

Some caveats:

- group theory: classification of finite simple groups had a gap for 25 years (e.g. around quasithin groups)
- number theory: geometric Langlands program (Gaitsgory et al)
- differential geometry: foundations of symplectic geometry, virtual transversality techniques

Lean and Mathlib

Lean is an interactive theorem prover.

Mathlib is its mathematical library

- 290 000 theorems about 138 000 definitions (many automatically generated)
- 772 contributors
- 60 reviewers, among which 28 maintainers who accept contributions
- All kinds of math: a monolithic library, because any part of math could be useful in combination with any other, and different parts of the library should be compatible.

What has been formalised already: let's guess

- Banach–Schauder open mapping theorem
- Birkhoff Ergodic Theorem
- Mandelbrot set is connected
- Cauchy-Kovalevskaya Theorem on existence of an analytical solution of an analytical PDE
- Denjoy's theorem: a C^2 orientation-preserving diffeomorphism of the circle with an irrational rotation number is conjugate to a rotation
- Sphere eversion
- Existence of Haar measure
- Existence of a smooth partition of unity
- Feit–Thompson theorem/odd order theorem
- Fermat's Last Theorem
- Four colour theorem
- Freyd–Mitchell embedding theorem
- Galois correspondence
- Herman-Yoccoz theorem on linearization of a circle diffeomorphism
- Jordan curve theorem
- Liouville theorem: an entire holomorphic function is a constant
- Hilbert's Nullstellensatz
- Picard-Lindelöf theorem (existence and uniqueness of solutions of ODEs)
- Poincaré-Bendixson Theorem
- Poincaré recurrence theorem
- Prime number theorem
- Sard's Theorem
- The continuum hypothesis is independent of ZFC.

Let's guess: the answer

Only two are not formalised yet (AFAIK)

- Denjoy's theorem on rotation number
- Herman–Yoccoz theorem on linearization of a circle diffeomorphism

Notable formalisation projects (pre-AI)

2005 Four colour theorem

2012 Odd Order Theorem

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project
- 2023 Upper bound on diagonal Ramsey numbers

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project before referee report
- 2023 Upper bound on diagonal Ramsey numbers before referee report

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project before referee report
- 2023 Upper bound on diagonal Ramsey numbers before referee report
- 2023 Marton's polynomial Freiman-Rusza conjecture (Tao et al)

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project before referee report
- 2023 Upper bound on diagonal Ramsey numbers before referee report
- 2023 Marton's polynomial Freiman-Rusza conjecture (Tao et al)
took 3 weeks; complete before paper submitted

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project before referee report
- 2023 Upper bound on diagonal Ramsey numbers before referee report
- 2023 Marton's polynomial Freiman-Rusza conjecture (Tao et al)
took 3 weeks; complete before paper submitted
- 2025 Disproof of the Aharoni–Korman conjecture (Mehta)
- 2025 Carleson's theorem

Notable formalisation projects (pre-AI)

- 2005 Four colour theorem
- 2012 Odd Order Theorem
- 2014 Kepler's conjecture (Hales et al)
- 2019 Ellenberg-Gijswijt's result on the cap set conjecture
- 2022 Liquid Tensor Experiment (Commelin et al):
fundamental lemma about condensed mathematics
- 2022 Unit fractions project **before referee report**
- 2023 Upper bound on diagonal Ramsey numbers **before referee report**
- 2023 Marton's polynomial Freiman-Rusza conjecture (Tao et al)
took 3 weeks; complete before paper submitted
- 2025 Disproof of the Aharoni–Korman conjecture (Mehta)
- 2025 Carleson's theorem
- 2026 Viazovska's 8d sphere packing (**partially AI**)
- 2026 Prime Number theorem (classical and modern error bounds)
(**partially AI**)

Notable AI formalization projects

- Jordan–Schönflies theorem
- Bieberbach conjecture (de Branges' theorem)
- Conway–Schneeberger 15 theorem
- Odd Order theorem
- Green–Tao theorem (primes in arithmetic progressions)
- Fermat's Last Theorem (AI formalization; also human-led project)
- Poincaré conjecture/Perelman's theorem
- current projects include the Stacks project, the classification of finite simple groups

AI formalization of current research

- Counterexample to the Erdos unit distance conjecture
- blow-up for forced Navier–Stokes equations
- two weeks ago: openAI releases claimed solutions to hundreds of open problems, about 40% with (partial) formalisation

State of AI formalisation

- autoformalisation (AI turns a paper into Lean code) is getting easy
- **careful**: does the Lean statement match the English mathematics?
stupid example: junk values, $2 - 3 = 0$ over $\mathbb{N}$
- Does the Lean proof match the prose write-up? Recent example:

NL, equation (10.19): $\left| \int \pi w \cdot \nabla \chi_R \right| \leq \frac{C_T}{R} \left[(B_R + 1) B_R^{1/2} + R^{-3/4} B_R^{3/4} \right], \text{ for a.e. } t \in (0, T), \quad (3.4)$

Lean estimate: $\left| \int \pi w \cdot \nabla \chi_R \right| \leq C_T \left[(B_R^{1/2} + 1) \left(\frac{A_R}{R} + \frac{1}{R^2} \right) + R^{-7/4} B_R^{3/4} \right], \forall t \in (0, T), \quad (3.5)$

Bastounis, Circelli, Hansen 2026: Navier–Stokes lost in translation.

State of AI formalisation (cont)

- does the Lean code “cheat” (e.g. meta-programming abuse)?
can check using proper tooling (**comparator**)
- trusted mathematical definition?
solution: **expert audit; build shared libraries**
- is the code maintainable? scalable? can it be re-used?
AI code falls very short here

Section 2

Mathlib and the formalization ecosystem

Why formalize?

- mathematical formalization has the potential to revolutionize mathematics
- the future of AI for mathematics lies in formalization
- Formalization is fun — like a video game or programming
- Because of the side effects: it can positively transform mathematics

Formalization can change mathematics



verified proofs, down to the axioms
reviewing: focus on importance, techniques and presentation



enable massive collaboration



understanding: expand any detail



creation: faster modification and generalisation

AI-generated mathematics is only trustworthy when formalised.

Understanding and creation

- understanding: reader chooses amount of detail

Demo by Patrick Massot and Kyle Miller:

[https://kmill.github.io/informalization/
ContinuousFrom.html](https://kmill.github.io/informalization/ContinuousFrom.html)

Understanding and creation

- understanding: reader chooses amount of detail

Demo by Patrick Massot and Kyle Miller:

[https://kmill.github.io/informalization/
ContinuousFrom.html](https://kmill.github.io/informalization/ContinuousFrom.html)

- creation: can this lemma be generalised? unused assumptions?
- database of theorems: searching known and related results

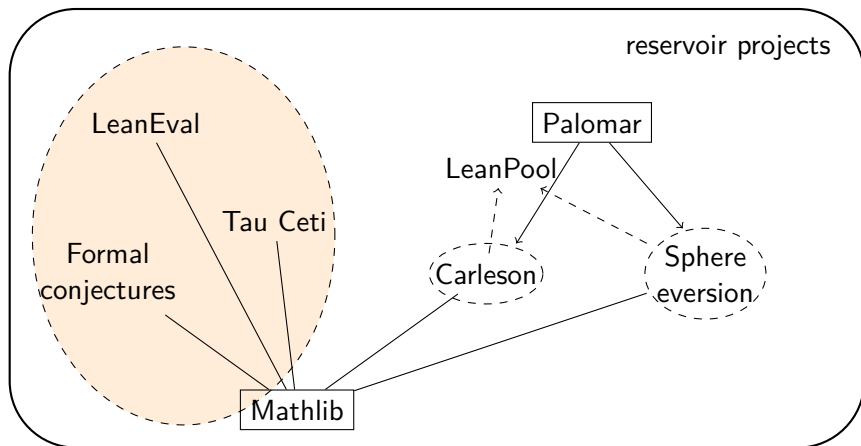
Community and collaboration

Lean checks the proof correctness → enables **large scale collaboration**.

Example: Carleson project

- Carleson: Fourier series of $f \in L^2$ converges pointwise a.e. to f
- subtle statement; all known proofs are hard and technical
- formalize modern generalization by Thiele et al to doubling metric measure spaces
- **28 contributors**, many undergraduates or not analysts
- a website, a github repository, a blueprint, a dependency graph, a Zulip channel
- a detailed $\text{\LaTeX}$ proof (“blueprint”)
- 35–40k lines of new Lean code
- many basics in harmonic analysis, suitable for Mathlib

Lean's formal math ecosystem



Lean's formal math ecosystem

- **Reservoir**: registry for “all” Lean packages
- **LeanEval**: (autoformalization system) benchmark of hard formalization problems. Solved problems with short statements; submitted solutions stay secret.
- **Formal conjectures**: repository of formalized mathematical conjectures
- **Tau Ceti**: AI-generated library of formal mathematics (building on and following mathlib)

Lean's formal math ecosystem

- **Reservoir**: registry for “all” Lean packages
- **LeanEval**: (autoformalization system) benchmark of hard formalization problems. Solved problems with short statements; submitted solutions stay secret.
- **Formal conjectures**: repository of formalized mathematical conjectures
- **Tau Ceti**: AI-generated library of formal mathematics (building on and following mathlib)
- **LeanPool**: monorepository collecting “significant” projects outside of mathlib
“AFP for Lean” — duplicate or conflicting definitions possible
- **Palomar** indexes significant mathematical formalizations: clear main statements, robustly audited
Could become the arXiv for mathematical formalizations.

Lean's formal math ecosystem

- **Reservoir**: registry for “all” Lean packages
- **LeanEval**: (autoformalization system) benchmark of hard formalization problems. Solved problems with short statements; submitted solutions stay secret.
- **Formal conjectures**: repository of formalized mathematical conjectures
- **Tau Ceti**: AI-generated library of formal mathematics (building on and following mathlib)
- **LeanPool**: monorepository collecting “significant” projects outside of mathlib
“AFP for Lean” — duplicate or conflicting definitions possible
- **Palomar** indexes significant mathematical formalizations: clear main statements, robustly audited
Could become the arXiv for mathematical formalizations.

Mathlib is the foundation for all of these.

Why mathlib? A human-curated community project

- Build trust in mathematical literature: audited definitions and statements with formal proof
- modern Bourbaki: be the **last** to formalise a given topic
- human understanding: canonisation of mathematics
- research project: discover formalization best practices
inspires mathematical generalisations, design lessons,
engineering insight
- community building: connection and collaboration across areas
of mathematics
- shared infrastructure: vetted definitions and designs to benefit
everybody

Summary

- 1 Formalisation is fun and has the potential to transform mathematics.
- 2 Autoformalisation at scale is getting possible.
- 3 Curated formalisations remain valuable, and offer insights into both mathematics and mathematical formalisation design.
- 4 Mathlib is critical infrastructure for formalised mathematics.

Summary

- 1 Formalisation is fun and has the potential to transform mathematics.
- 2 Autoformalisation at scale is getting possible.
- 3 Curated formalisations remain valuable, and offer insights into both mathematics and mathematical formalisation design.
- 4 Mathlib is critical infrastructure for formalised mathematics.

Thanks for listening! Any questions?

Bonus slides

- ▶ How review somebody else's formalization
- ▶ Scope of mathlib
- ▶ Mathlib governance/how are decisions made

Reviewing somebody else's formalization

- Be cautious about building it locally
- Double-check the main statement:
 use mathlib or formal conjectures statements if possible
- Check definitions particularly carefully;
 use mathlib ones if you can
- Checking good-faith proofs: `#print axioms foo` should say
 (a subset of) `propext`, `Classical.choice`, `Quot.sound`
- Fully robust checking: use [comparator](#)
 Create a Challenge file with your main statement;
 your proof is verified against that.

“Have you submitted to Palomar yet?”

What do we review? The scope of mathlib

- “all areas of mathematics”, of course!

What do we review? The scope of mathlib

- “all areas of mathematics”, of course! Right?
- really everything? just research? just current research?
controversial topic

What do we review? The scope of mathlib

- “all areas of mathematics”, of course! Right?
- really everything? just research? just current research?
controversial topic
- practical considerations: maintainer expertise? review capacity?
- case studies: combinatorial game theory, PhysLean, CSLib, Mathlib/Computability

Governance around mathlib: who makes the decisions?

- large distributed project:
much communication is asynchronous, via text
- different community teams: admin, webpage, CI admins, code of conduct, mathlib maintainers (and reviewers)
Details at
<https://leanprover-community.github.io/teams.html>
- separate entities: Lean FRO; mathlib initiative

Governance around mathlib: who makes the decisions?

- large distributed project:
much communication is asynchronous, via text
- different community teams: admin, webpage, CI admins, code of conduct, mathlib maintainers (and reviewers)
Details at
<https://leanprover-community.github.io/teams.html>
- separate entities: Lean FRO; mathlib initiative
- technical leadership: maintainers team
- ML community coordinator to help coordinate this

Decision-making in mathlib

- try to avoid majority votes, consensus-based decision making (if possible)
- involve broader community when possible — almost all technical discussions happen in the open!
- “serving leadership” — lots of listening, empathy, looking for positive-sum outcomes
- Community coordinator: yours truly

Challenge: dealing with AI slop

AI slop projects posted on zulip

LLM-written Zulip posts (forbidden!)

LLM-generated PRs (without good understanding)

using LLM to respond to review comments

Problems of AI slop

AI slop projects posted on zulip; LLM-written comments

LLM-generated PRs (without good understanding)
using LLM to respond to review comments

Can be well-meant, but risks burning out reviewers and maintainers

Quality bar: AI-generated code is not good enough

signal to noise ratio on zulip

often invisible as moderated!

Current solutions

- moderation tooling on zulip
- AI policy: **must not** use AI to write comments on Zulip or github
- AI policy: AI use must be disclosed on the PR
- label LLM-generated to highlight LLM-generated PRs
- further policy changes under discussion