

Formalisation of mathematics: insight into an ongoing revolution

Michael B. Rothgang (he/him)

Formalised mathematics group, Universität Bonn



Data Science and Machine Learning seminar, FSU
October 9, 2026

Outline of today's talk

- 1 Lean and mathematical formalization
- 2 Mathlib and the formalization ecosystem
- 3 Differential geometry in mathlib
- 4 Manifolds for mathlib
- 5 Challenges for formalising differential geometry

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.

How does it work?

You want to prove a theorem in Lean.

- You import other definitions and theorems you need from a library
- You write a statement in the Lean language
- You write a proof, using tactics to help you
- The Lean engine checks that your proof is correct

Now your theorem is part of the library, and can be used in other proofs.

How does it work?

You want to prove a theorem in Lean.

- You import other definitions and theorems you need from a library
- You write a statement in the Lean language
- You write a proof, using tactics to help you
- The Lean engine checks that your proof is correct

Now your theorem is part of the library, and can be used in other proofs.

Some other interactive theorem provers:

Rocq, Isabelle/HOL, Agda, Metamath, Mizar. . .

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
- 3 days 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:

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

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

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
only requires statements of main results

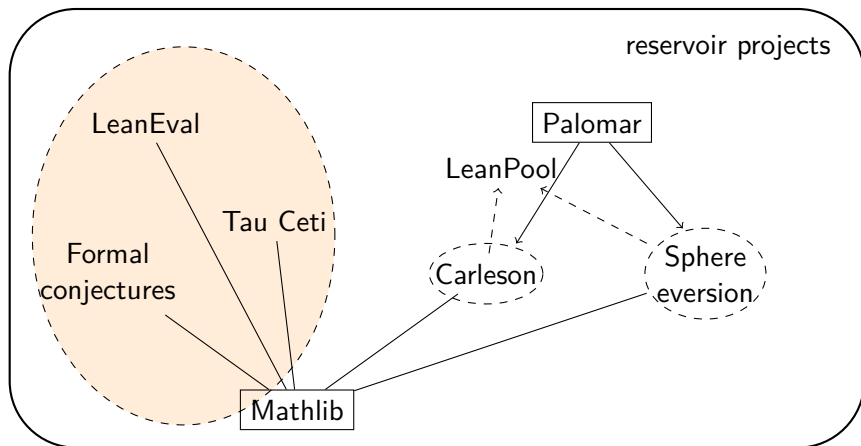
Community and collaboration

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 for each area
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

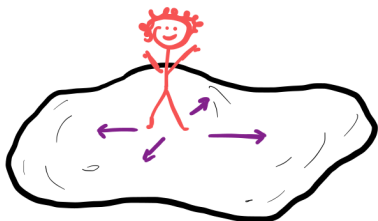
Section 3

Differential geometry in mathlib

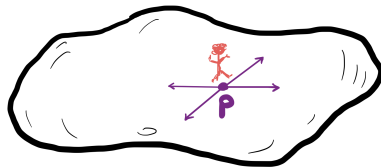
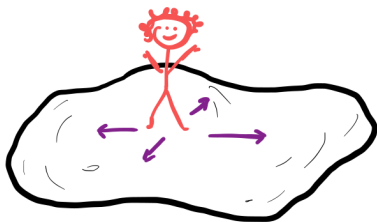
Differential geometry...

... is the study of smooth manifolds and their Lie group, Riemannian, symplectic, foliations, vector bundle, ... structures

What is a manifold?



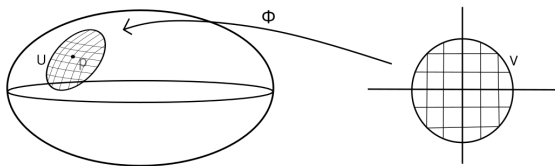
What is a manifold?



surface of a potato is a manifold: locally looks like a disk

Smooth manifolds

- topological **manifold**: second countable Hausdorff topological space M locally homeomorphic to open ball in $\mathbb{R}^n$
- every $p \in M$ has a coordinate chart: $p \in U \subset M$ open, homeomorphism $\phi: V \rightarrow U$ for $V \subset \mathbb{R}^n$ open ball
- **smooth manifold**: all coordinate transformations from overlapping charts are smooth



Picture courtesy of Dominik Gutwein.

Examples of smooth manifolds

- empty set (of any dimension)
- 0-dimensional: isolated points
- 1-dimensional: $\mathbb{R}$, S^1
- n -dimensional: open disc $\mathbb{D} \subset \mathbb{R}^n$
- $n = 2$: $\mathbb{R}^2$, S^2 , T^2 , Σ_g for $g \geq 1$



Examples of smooth manifolds

- empty set (of any dimension)
- 0-dimensional: isolated points
- 1-dimensional: $\mathbb{R}$, S^1
- n -dimensional: open disc $\mathbb{D} \subset \mathbb{R}^n$
- $n = 2$: $\mathbb{R}^2$, S^2 , T^2 , Σ_g for $g \geq 1$



- $n \geq 3$: complicated; classification for $n \geq 4$ impossible
- $\mathbb{R}^n$, S^n , T^n , $\mathbb{RP}^n$, $\mathbb{CP}^n$,
 $\{[z_0 : z_1 : z_2 : z_3 : z_4] \in \mathbb{CP}^4 \mid z_0^5 + \dots + z_4^5 = 0\}$
configuration spaces in physics and engineering
- **not** a manifold: letter “X”

Manifolds with boundary or corners

- $\overline{D} \subset \mathbb{R}^n$ is a manifold with **boundary**
- interior points locally look like open ball in $\mathbb{R}^n$,
boundary points look like open ball in upper half of $\mathbb{R}^n$
- manifolds with **corners**: local model is Euclidean quadrant
 $[0, 1]^n \subset \mathbb{R}^n$ has corners

Constraints for defining manifolds in Mathlib

- Meta-goal: generality should suffice for all of modern research

Constraints for defining manifolds in Mathlib

- Meta-goal: generality should suffice for all of modern research
- allow different smoothness (C^0 , C^k , smooth, analytic)
- allow boundary and corners
- allow different base field: $\mathbb{R}$, $\mathbb{C}$ or p -adic numbers $\mathbb{Q}_p$
- infinite-dimensional manifolds (e.g. $C^k(M, N)$)

This definition is more general than any mathematics textbook I know.

Formalising manifolds with boundary

A smooth manifold includes several data:

- M : the manifold (e.g. $\overline{\mathbb{D}}$)
- H : the local model, a topological space (e.g. $\mathbb{H}$)

Formalising manifolds with boundary

A smooth manifold includes several data:

- M : the manifold (e.g. $\overline{\mathbb{D}}$)
- H : the local model, a topological space (e.g. $\mathbb{H}$)
- E : normed space (e.g. $\mathbb{R}^2$)
- I : model with corners, continuous map $H \rightarrow E$
(e.g. **canonical inclusion**)

Formalising manifolds with boundary

A smooth manifold includes several data:

- M : the manifold (e.g. $\overline{\mathbb{D}}$)
- H : the local model, a topological space (e.g. $\mathbb{H}$)
- E : normed space (e.g. $\mathbb{R}^2$)
- I : model with corners, continuous map $H \rightarrow E$
(e.g. **canonical inclusion**)
- charts on M (one preferred chart at each point)
- compatibility condition: transition maps lie in structure groupoid

Overview: differential geometry in mathlib

- smooth manifolds, smooth maps, (continuous) differentiability
- (manifold) Fréchet derivative, chain rule
- diffeomorphisms, local diffeomorphisms; smooth immersions and embeddings
- products and disjoint unions of manifolds
- classification of 0-dimensional manifolds
- examples: $\mathbb{R}^n$, half-space, quadrants; intervals
unit sphere; units in Lie groups

Overview: differential geometry in mathlib (cont.)

- (topological and smooth) vector bundles
- basic constructions: trivial bundle, direct sum, product bundle, hom bundle, (co)tangent bundle
- (continuous) differentiability of sections, smooth bundle maps
- Lie bracket of vector fields; Lie groups and their Lie algebra
- smooth bundle metrics; (basic) Riemannian manifolds
- covariant derivatives
- existence and uniqueness of Levi-Civita connection
- existence of integral curves and local flows
- smoothness of local flows

What's missing?

- smooth submanifolds (in progress); sub-bundles
- quotients of manifolds; gluing (both in progress)
- implicit and inverse function theorems (in progress)
- constant rank theorem; regular value theorem (in progress)
- differential forms (in progress)
- classification of 1-manifolds and 2-manifolds
- smooth fibre bundles
- jet bundles

More help writing high-quality code is welcome!

Section 5

Challenges for formalising differential geometry

High-level challenges

1 boilerplate: typeclasses

“let E be a smooth vector bundle over a smooth manifold M ”

```
variable (F : Type*) [NormedAddCommGroup F] [NormedSpace ℝ F]
(E : M → Type*) [TopologicalSpace (TotalSpace F E)]
[∀ x, AddCommGroup (E x)] [∀ x, Module ℝ (E x)] [∀ x : M, TopologicalSpace (E x)]
[FiberBundle F E] [VectorBundle ℝ F E] [ContMDiffVectorBundle n F E I]
```

2 verbosity: “let $s : M \rightarrow E$ be a C^n section at x ”

```
variable {s : (x : M) → V x} {x : M}
| {hs : ContMDiffAt I (I.prod J(ℝ, F)) n (fun x ↦ TotalSpace.mk' F x (s x)) x}
```

3 invisible mathematics (cf. Andrej Bauer)

use subtypes, or junk value pattern:

lots of trivial proofs “this point lies in this open set”

High-level challenges

- 4 Can we find better abstractions?
ContinuousWithinAt, ContMDiffWithinAt are a good abstraction, but very low-level
- 5 Better abstractions: can we abstract “this is just a local argument”? Make a tactic for this?

High-level challenges

- 4 Can we find better abstractions?
ContinuousWithinAt, ContMDiffWithinAt are a good abstraction, but very low-level
- 5 Better abstractions: can we abstract “this is just a local argument”? Make a tactic for this?
- 6 API consistency vs. combinatorial explosion
 - often 4 variants of each lemma
ContMDiff{,On,At,WithinAt}
 - mirror for MDifferentiable*, ContDiff*, Differentiable*
(and sometimes Continuous*) $\Rightarrow$ **20 versions**
 - applied vs non-applied: contMDiffAt_smul vs
contMDiffAt_fun_smul $\Rightarrow$ up to **40 versions**
 - real-life example: one PR (50 lines) became 5 (250 lines)
 - real-life example: 1000 lines of mathlib code, just copy-pasting
ContMDiff lemmas to MDifferentiable

Some solutions

typeclasses: proposed solution (improved **class** abbrev),
needs changes to Lean itself

Some solutions

typeclasses: proposed solution (improved **class** abbrev),
needs changes to Lean itself

write custom elaborators to reduce the verbosity

- auto-convert section into non-dependent function

`T% s` means **fun** `x => TotalSpace.mk' F x (s x)`

Some solutions

typeclasses: proposed solution (improved **class** abbrev),
needs changes to Lean itself

write custom elaborators to reduce the verbosity

- auto-convert section into non-dependent function
 $T\% s$ means **fun** $x \Rightarrow \text{TotalSpace.mk}' F x (s x)$
- infer model with corners:
 $\text{CMDiffAt } n f x$ means $\text{ContMDiffAt } I J n f x$
- vision: almost never need to write the model with corners
- optional: if needed, can write code as before

for example: “ X is a C^n vector field at x ” becomes
 $(hX: \text{CMDiffAt } (T\% X) x)$

Some solutions (cont.)

Use tactics to auto-generate and keep APIs in sync

idea: `to_mdifferentiable` attribute

- replace `ContMDiff*` (hypotheses and goals)
by the analogous `MDifferentiable` statement
- implementation very similar to `to_additive`

Some solutions (cont.)

Use tactics to auto-generate and keep APIs in sync

idea: `to_mdifferentiable` attribute

- replace `ContMDiff*` (hypotheses and goals)
by the analogous `MDifferentiable` statement
- implementation very similar to `to_additive`

`to_fun` attribute

given a lemma, automatically generate the applied form
(e.g. `contMDiff_smul` to `contMDiff_fun_smul`)

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

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

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