ICMAT: GROUP THEORY SEMINAR

Introduction to formalization in Lean

From writing math to writing mathlib:

an intro for beginners, by a beginner.

Hang Lu Su

Math PhD, ICMAT | MadLean organizer

May 21, 2026

Outline

- What is Lean / mathlib?
  - Demo
- What am I doing at a Group Theory seminar?
  - How Lean is relevant to mathematicians
- Leaning in into Lean
  - Technicalities of the language
- How I contributed to mathlib.
  - Formalizing finitely presented groups
- How **you** can contribute to mathlib.

SECTION

What is Lean / mathlib?

Lean: formal verification language with interactive proof assistant

Let's look at the a simplied version of the official live demo.

- In Lean, a proposition is represented as a type `P`, and a proof of that proposition is a term `t` inhabiting that type. So to prove a theorem with statement `P`, Lean must construct a term `t` such that `t : P`.

```lean
theorem true_is_true : True := by
  exact True.intro
```

LEAN

- Lean has a nice InfoView with the tactic state and current goal.

> *"Doing math without tactic states is like doing chess without a chessboard."*
>
> – Patrick Massot (this is who I heard it from.)

Mathlib: library of formalized mathematics in Lean

Mathlib: monolithic library of definitions and theorems built on top of Lean.

- Open-source: anyone can access the code, anyone can pull request.
- Official discussion channel is on Zulip.
- Named for easy search.
- Optimized for re-usability of definitions and lemmas.
- Can be imported for separate formalization projects.

```
import Mathlib

-- Your formalization starts here!
```
LEAN

SECTION

Why am I here?

Isn't this a Group Theory seminar?

How is this relevant to your life?

Where I was when mentally I finished my PhD at ICMAT

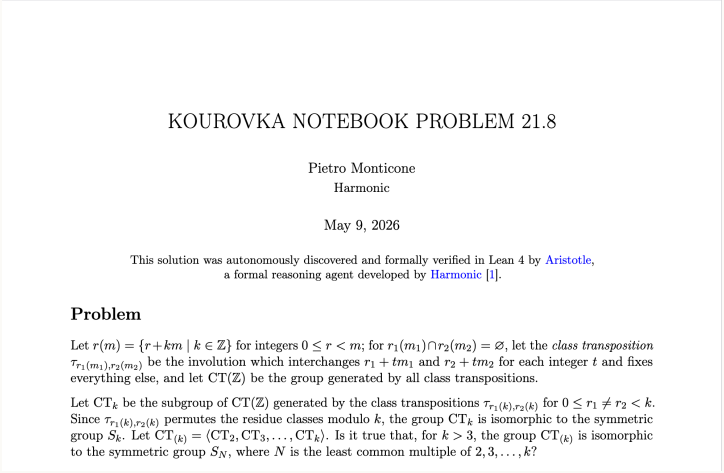
- Lean and mathlib got a lot of prominence.
- ChatGPT started getting good.
  - I received corrections for my thesis. Calculus 1 type error (I forgot calc).
  - I asked ChatGPT-o1 to help me fix and to my surprise it did it really well!

What will the future of math look like?

- Main problem with LLMs: they hallucinate a lot.
- Lean statements are automatically type-checked for correctness.
- If humans can read/write Lean statements, LLMs could generate proofs.
- I registered for ItaLean2025: Bridging Formal Mathematics and AI conf.

Couple months later, the future is already now.

LLM solves problems in 21st ed. of Kourovka's notebook



KOUROVKA NOTEBOOK PROBLEM 21.8

Pietro Monticone
Harmonic

May 9, 2026

This solution was autonomously discovered and formally verified in Lean 4 by Aristotle, a formal reasoning agent developed by Harmonic [1].

Problem

Let $r(m) = \{r + km \mid k \in \mathbb{Z}\}$ for integers $0 \le r < m$; for $r_1(m_1) \cap r_2(m_2) = \varnothing$, let the *class transposition* $\tau_{r_1(m_1),r_2(m_2)}$ be the involution which interchanges $r_1 + tm_1$ and $r_2 + tm_2$ for each integer t and fixes everything else, and let $\mathrm{CT}(\mathbb{Z})$ be the group generated by all class transpositions.

Let CT_k be the subgroup of $\mathrm{CT}(\mathbb{Z})$ generated by the class transpositions $\tau_{r_1(k),r_2(k)}$ for $0 \le r_1 \ne r_2 < k$. Since $\tau_{r_1(k),r_2(k)}$ permutes the residue classes modulo k, the group CT_k is isomorphic to the symmetric group S_k. Let $\mathrm{CT}_{(k)} = \langle \mathrm{CT}_2, \mathrm{CT}_3, \ldots, \mathrm{CT}_k \rangle$. Is it true that, for $k > 3$, the group $\mathrm{CT}_{(k)}$ is isomorphic to the symmetric group S_N, where N is the least common multiple of $2, 3, \ldots, k$?

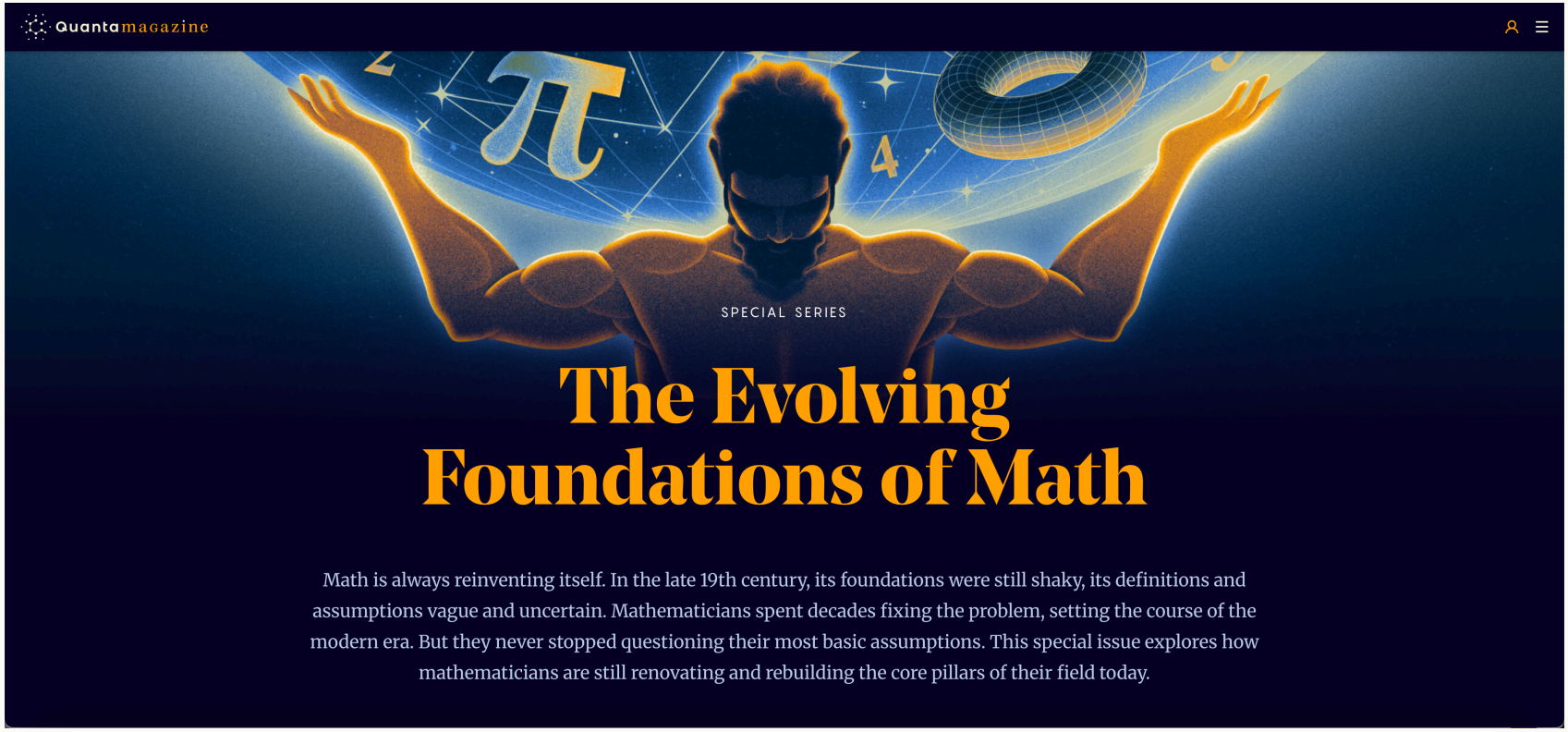
- **Pietro Monticone** uploads solutions 20.125, 21.8, 21.24, and 21.150
- Solutions have been independently generated by LLM Aristotle (Harmonic).
- Links to formally verified solution provided in live Lean editor (ex: 21.8 ; gist)
- Could not find method, but P. Monticone has a thread about solution to Erdös problem #650 using ChatGPT-5.4 and Aristotle on Bluesky.

A few high-profile projects and evangelists

- **Kevin Buzzard:** Fermat's Last Theorem project.
  - Easy to state, hard to prove. Builds infrastructure for computer-assisted mathematics, easily crowd-sourceable collab, and educational. See post.
- **Peter Scholze:** Liquid Tensor Experiment.
  - Lean can certify Fields medalist level work, including proofs the author is unsure about, such as for Theorem 1.1 of this Xena blog post.
- **Terry Tao:** Equational Theories Project.
  - Test project for veracity of 22028942 magma equational laws implications. By crowdsourcing and automation, 99.99% completed in 3 weeks.

Media coverage

Quanta Magazine did a series on math foundations this year.



Best way to start

- Natural Number Game. Construct natural numbers from Peano axioms!
- Mathematics in Lean. Textbook with exercises in Lean.
- Lean Zulip #new members. 24/7 internet hotline, helpers from all timezones.

- MadLean Meetups. Weekly seminars/workshops at UCM.
  - I co-founded them, so you know at least one person :).
  - First guest seminar next week, from the organization writing Lean itself!

May 27th, 18:00 Facultad de Matemáticas, UCM aula 117

Emilio Jesús Gallego Arias (Lean FRO) will talk about Verso Blueprint.

SECTION

How I contributed to mathlib.

My first formalization target

THEOREM: Reidemeister-Schreier theorem

If G is finitely presented and H is a finite-index subgroup of G, then H is finitely presented. See p. 169-175 of my thesis.

- Spent a week stuck understanding the *meaning* of the algebraic proof.
- The geometric insights helped me prove my favorite doctoral results.
- **Finitely presented groups were not formalized in mathlib!**

WHAT I LEARNED

Target theorem helps build useful API by working backwards.

SUBSECTION

Battle of definitions

Math writing vs mathlib: a culture shock.

Formalizing finitely presented groups

DEFINITION: Finitely presented group

A group G is **finitely presented** if there exist a **finite** set of generators S and a **finite** set of relations R such that

$$G \cong \langle S \mid R \rangle := F(S)/\langle\langle R \rangle\rangle$$

where $F(S)$ is the free group on S and $\langle\langle R \rangle\rangle$ is the normal closure of R.

Main issue: there are many valid ways to formalize this!

Going from set theory to type theory wasn't trivial either.

Using a surjection (due to Riccardo Brasca)

DEFINITION: Finitely presented group

$$\exists S, R \text{ finite such that } G \cong F(S)/\langle\langle R \rangle\rangle$$

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (n : ℕ) (f : FreeGroup (Fin n) →* G),
    Function.Surjective f ∧ IsNormalClosureFG f.ker
```

LEAN

- S is `Fin n`, $\langle\langle R \rangle\rangle$ is `f.ker`
- "`Fin n` *is a natural number i with the constraint that i < n. It is the canonical type with n elements.*" docs: Init/Prelude/Fin

Starting def at ItaLean. In number theory, FP groups special case of FG groups.

Using a surjection - Part 2

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (n : ℕ) (f : FreeGroup (Fin n) →* G),
    Function.Surjective f ∧ IsNormalClosureFG f.ker
```
LEAN

Feedback from project teammates: (also mathematicians and beginners)

- **1**: *"This isn't really what a finitely presented group is."*
- **2** *"Do you even know what a finitely presented group is?!"* (hyperbole)

Our code at the end of the workshop.

- Feels like the start of a nice independent project but not mathlib-ready.
  - There are no proofs, just definitions which is a no-no for new contributors.
- I decide to re-work it over Christmas break with the goal of upstreaming.

Using isomorphisms (mine)

DEFINITION: Finitely presented group

$$\exists S, R \text{ finite such that } G \cong F(S)/\langle\langle R \rangle\rangle$$

LEAN

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (α : Type) (_: Finite α) (rels : Set (FreeGroup α)) (_ : rels.Finite),
  Nonempty (G ≃* (PresentedGroup rels))
```

- S is `(α : Type) (_: Finite α)`, R is `rels : Set (FreeGroup α)`

Things felt smooth and easy-ish because that's what I was used to.

Using isomorphisms - Part 2

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (α : Type) (_: Finite α) (rels : Set (FreeGroup α)) (_ : rels.Finite),
  Nonempty (G ≃* (PresentedGroup rels))
```
LEAN

I wrote a lot of code with this!

- Code to go between the different definitions + isomorphism theorem.
- `FP_is_FG` instance, `FreeGroup` instance, trivial instance, mul $\mathbb{Z}$ instance, etc

USE OF AI

I used Aristotle, then GPT-5.2, 5.3 and 5.4 to assist and learned a lot.

Using isomorphisms - Part 3

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (α : Type) (_: Finite α) (rels : Set (FreeGroup α)) (_ : rels.Finite),
  Nonempty (G ≃* (PresentedGroup rels))
```
LEAN

Review: *"both the quantification over* `Type` *as wells as the* `Nonempty` *strike me as a little strange for the main definition."*- Source.

WHAT I LEARNED

- **Math:** *"a group is FP because ∃ isomorphism to FP group"* is fine.
- **Lean:** have to deal with a non-specific isomorphism and unspecific `Type` (or `Type*`) which can cause universe issues *as a default*.

Using isomorphisms - Part 4: Type digression

Universe hierarchy in CIC to avoid Girard's paradox [1] (the type-theoretic analogue of Russell's paradox: `Type : Type` is inconsistent).

- **Consequence:** everything needs to be grounded at a universe level.

WHAT I LEARNED

- It's enough to say "*for some* `Type`" in math, but not Lean. Demo.

```
    Group.FG G ↔ ∃ (α : Type*) (_ : Fintype α) (φ : FreeGroup α →* G),
Function.Surjective φ := by
```
LEAN

- `←` You're providing a type of arbitrary universe level.
- `→` Lean has to figure out the universe level with nothing to go on!

Long story short, Riccardo was right

```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (n : ℕ) (f : FreeGroup (Fin n) →* G),
    Function.Surjective f ∧ IsNormalClosureFG f.ker
```
LEAN

WHAT I LEARNED

Why was this definition good?

- No `Nonempty`, no `Type` or `Type*` universe shenanigans.
- Clean. Another iteration used a lot of coersions (unnatural to TT).
- Isomorphism-invariance proof still flows. Demo

SECTION

You can contribute to mathlib!

Collaboration workflow

Collaborating is surprising smooth using Lean!

- Zulip to coordinate people.
- Github to PR request.
- Once definition set, statements can be formulated by experienced users and completed by beginners as correctness is automatically checked!

```lean
theorem myTheorem : IsFoo Bar := by
  sorry
```

Where should you find such opportunity locally, you may ask?

Come to MadLean!!!

Collaborations within MadLean

- The Finitely Presented Group API in Mathlib.
  - After merging definition, MadLean participants rewrote the proofs.
  - Everyone involved was first-time contributor.
  - Would write filler PR. (Example.)

```
theorem my_old_statement := by
  sorry
```
LEAN

  - Volunteers then complete the statement and go through code-review. Ex.
- MadLean on Zulip
- The Finitely Presented Groups discussion thread on Zulip.
  - A lot of iterations went on, more than I can say in this talk.

Collaborate with me

This summer I will be at **Imperial College London** working on the Annals of Mathematics formalization project with **Kevin Buzzard**'s group.

- If we provide Lean statements to AI, how useful are they at actually autoformalizing frontier math?
- Main bottleneck are **definitions**.
- Want working mathematician's input on definitions.
  - Formalized math vs informal math defs differ, but want to get it right.
- Not much in GGT right now, merge definitions, get students involved and learn Lean together!

Come talk to me!

Big thanks to everyone who guided me

USE OF HUMAN INTELLIGENCE

- **ItaLean2025 organizers:** for giving me the space to start.
- **Riccardo Brasca:** for getting me started.
- **Kevin Buzzard:** for his continued guidance over Zulip and Github.
- **Nehal Patel:** for investigating beginner struggles with me.
- **Thomas Browning:** for reviewing and rewriting my PRs with me.
- **Bhavik Mehta:** for teaching me type polymorphism.
- **Michael Rothgang:** for tips and reviews.
- **Sidharth Hariharan** & **Auguste Poiroux:** for last minute talk prep help.

Thank you!

- Thank you **Andrei Jaikin** for inviting me and organizing these seminars.
- Thank **you** for listening!

References

[1] J.-Y. Girard, “Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur,” Doctoral dissertation, 1972.

[2] L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer, “The Lean Theorem Prover (System Description),” in *Automated Deduction – CADE-25*, Cham: Springer, 2015, pp. 378–388.

[3] The mathlib Community, “The Lean Mathematical Library,” in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, New York: Association for Computing Machinery, 2020, pp. 367–381.

[4] J. Avigad, L. de Moura, S. Kong, and S. Ullrich, *Theorem Proving in Lean 4*. 2024. [Online]. Available: https://lean-lang.org/theorem_proving_in_lean4/

[5] The Lean Prover Community, “Mathematics in Lean.” [Online]. Available: https://leanprover-community.github.io/mathematics_in_lean/

[6] The Lean Prover Community, “Natural Number Game 4.” [Online]. Available: https://adam.math.hhu.de/#/g/leanprover-community/nng4

[7] H. L. Su, “A toolbox for left-orders of low complexity.” [Online]. Available: https://arxiv.org/abs/2512.07035

[8] Imperial College London FLT Project, “Fermat's Last Theorem.” [Online]. Available: https://imperialcollegelondon.github.io/FLT/

[9] The Lean Prover Community, “The Liquid Tensor Experiment.” [Online]. Available: https://github.com/leanprover-community/lean-liquid

[10] T. Tao and collaborators, “Equational Theories Project.” [Online]. Available: https://teorth.github.io/equational_theories/

The end!