IMDEA SOFTWARE: Invited talk

A mathematician's perspective on Lean

From writing math to writing mathlib:

a beginner's point of view.

Hang Lu Su

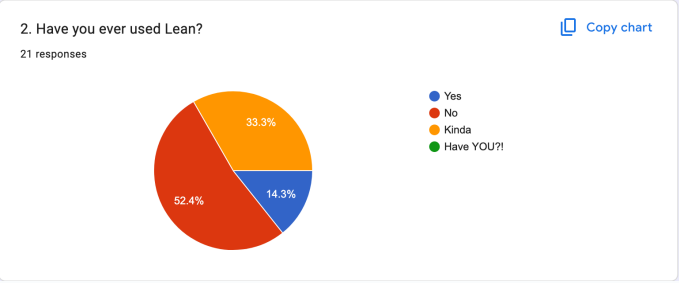
Math PhD, ICMAT | MadLean organizer

May 26, 2026

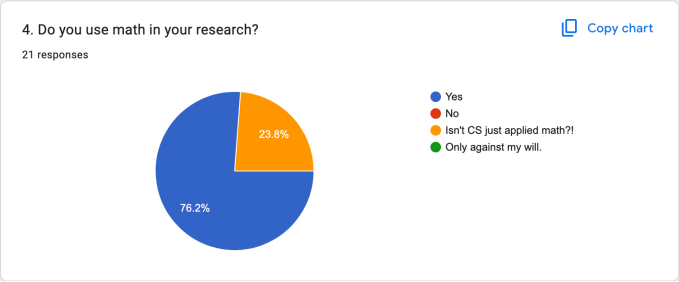
Thank you for the survey responses!

This is my first time giving a software talk so I made an anonymous survey.

1. Most of you have never used Lean.



2. All of you use math in your research.



What you wanted to hear about (+ what will be against your will)

1. Lean as a tool for checking mathematics.
   - Let's first do a demo.
   - Then talk about the techicalities of the language.
2. A broader discussion of why formalization matters.
   - Lean is becoming more and more relevant to mathematicians.
     - How the Lean/mathlib community relates to math, CS and AI.
3. How to get started.
4. Nobody asked for this but:
   - I will briefly talk about my formalization work and what's next.

SECTION

What is Lean / mathlib?

I'm impatient so we'll start with a demo.

Lean: formal verification language with interactive proof assistant

Let's look at the a simplified 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`.

```
theorem true_is_true : True :=
  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.
  - `exists_prime_factor` is called `Nat.exists_prime_and_dvd`.
- Optimized for re-usability of definitions and lemmas.
- Can be imported for separate formalization projects.

```
import Mathlib

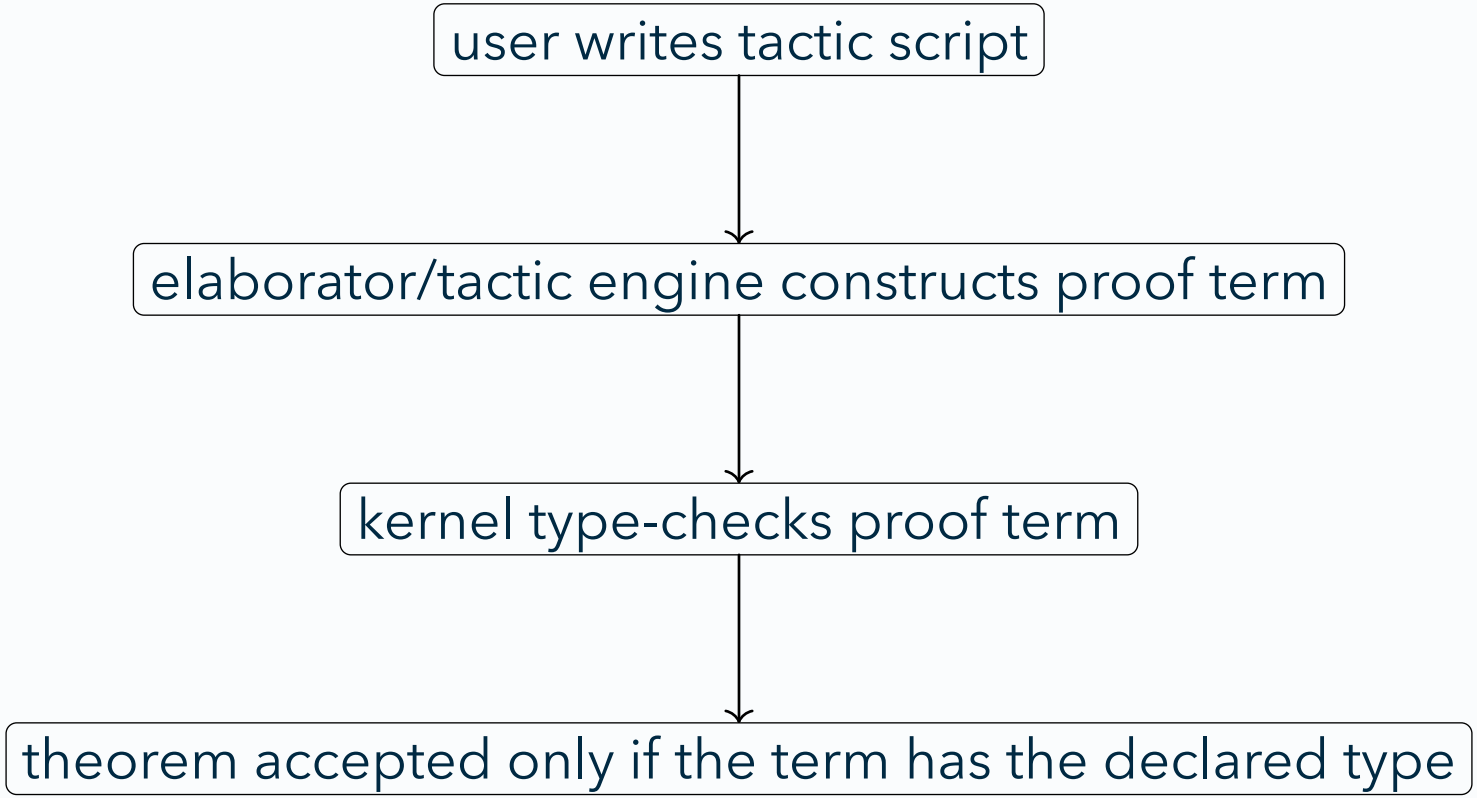
-- Your formalization starts here!
```
LEAN

SECTION

Let's Lean in!

Where I try to explain the technicalities of Lean to people who are infinitely more expert than me. #girlboss

How does Lean work?



Dependent type theory with Calculus of Inductive Constructions

- Type-based logic; dependent means the types can depend on input.
  - ```
    [1, 2, 3]       : Vector Nat 3
    [1, 2, 3, 4, 5] : Vector Nat 5
    ```
    LEAN
- Universe hierarchy in CIC to avoid Girard's paradox [1].
  - Russell's paradox: "the set of all sets" self-reference leads to contradiction.
  - Girard's paradox is the type theory analogue.
    - If `Type : Type`, then the "type of all types" is itself one of the types.
- Kernel typechecks, using rules for variables, universes, dependent functions, lambdas, application, conversion by computation, inductive types, and proof irrelevance. Source.
    ```

How is mathlib built on top of Lean?

These axioms are imported from Lean's standard `Init` environment for use.

- `propext`: if two propositions imply each other, treat them as equal.
- `Quot.sound`: Related representatives are equal in the quotient.
- `Classical.choice`: From a proof that a type is nonempty, choose an element.

From these, Lean derives:

- law of excluded middle: `P ∨ ¬ P`
- proof by contradiction: `¬¬ P → P`
- function extensionality: `(∀ x, f x = g x) → f = g`

Math theorems written on top of those imports.

SECTION

Why mathematicians care

Or how I ended up talking at a research software seminar.

Where I was when mentally I finished my PhD at ICMAT

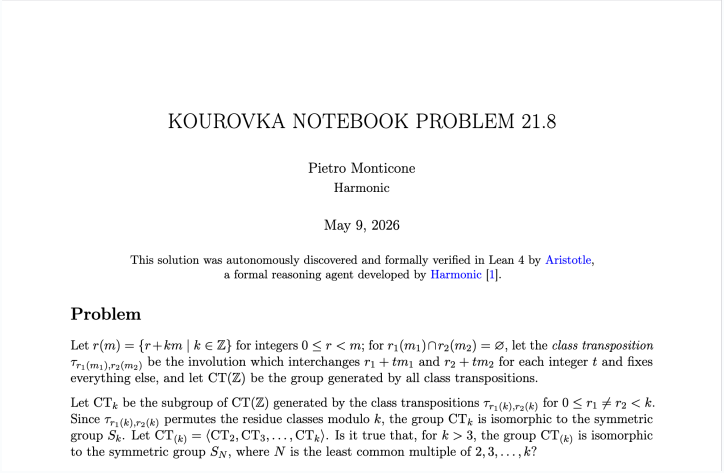
- Lean and mathlib got a lot of prominence.
- ChatGPT started getting good.
  - I received corrections for my thesis. Calculus 1 type error (I forgot calc).
  - I asked ChatGPT-o1 to help me fix and to my surprise it did it really well!

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 Aristotle (Harmonic)
  - Formally verified solutions provided in live Lean editor (ex: 21.8 ; gist)
- **Marc Lackenby** provides sol 21.10 using AI co-mathematician (DeepMind).

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. post ; repo
- **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 ; repo
- **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. web
- **Alex Kontovorich:** teaching first Analysis course as a Lean game.

Why is Lean popular over other languages?

Technical answer:

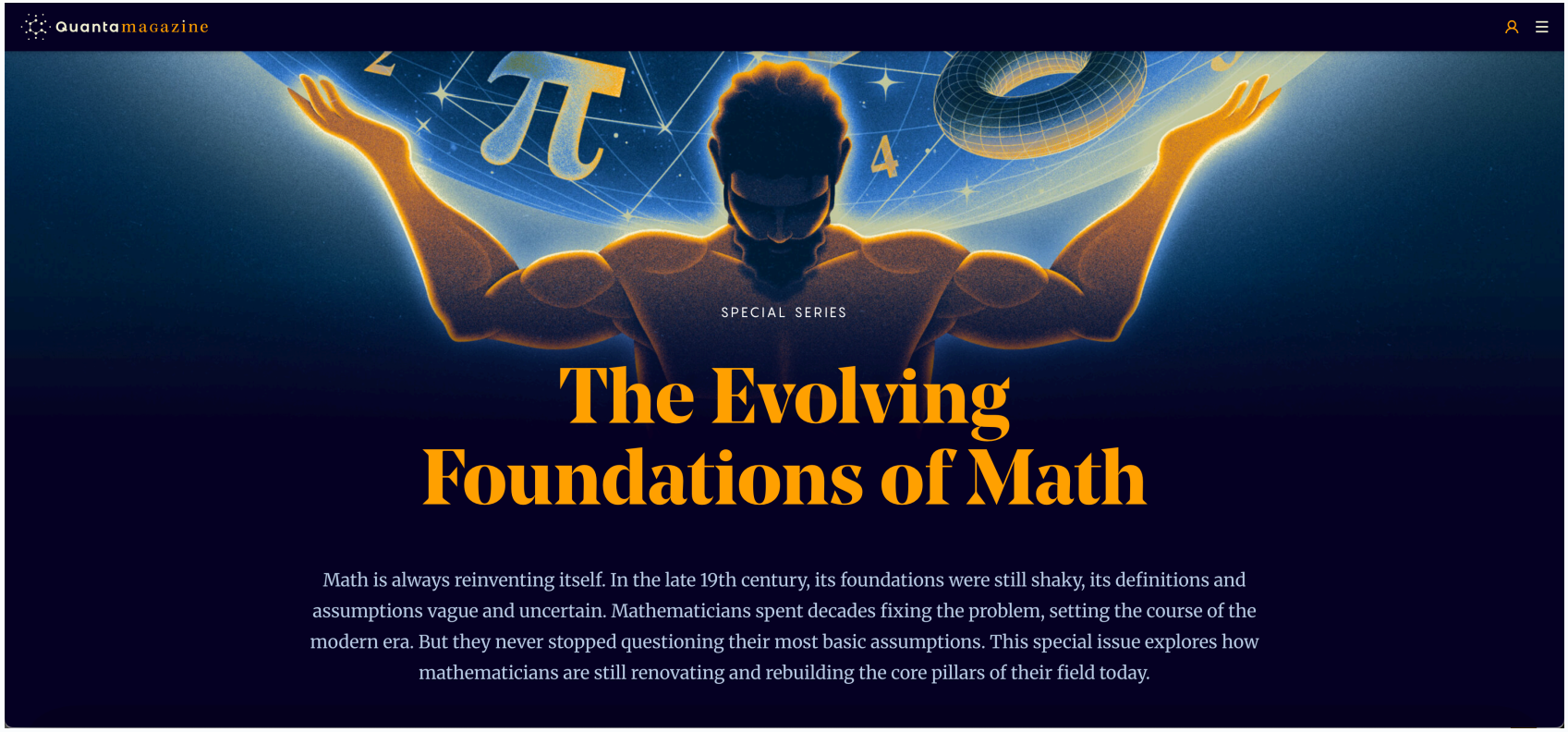
- dependent type theory gives expressivity
- elaborator hides much of the type-theoretic bureaucracy
- tactic system lets users build domain-specific automation
- compiler makes that automation fast enough
- mathlib turns isolated proofs into reusable infrastructure.

Honest answer: I don't know, just like idk why LaTeX is popular and standard.

- I've seen friends use Rocq 10 years ago and it seemed tedious. I nope'd.
- People report Lean is less tedious, so I gave it a try. NNG4 was so fun!
- Monolithic library with big profile projects makes it exciting!

Media coverage

Quanta Magazine did a series on math foundations this year.



Best ways 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.

Finitely presented groups are essential in infinite group theory

DEFINITION: Free group

Given a set S of **generators**, form the **alphabet** $S \cup S^{-1}$ (each generator plus a formal inverse). The free group $F(S)$ is:

- **Elements:** finite strings over $S \cup S^{-1}$, quotiented by the rewrite rules $ss^{-1} \to \varepsilon$ and $s^{-1}s \to \varepsilon$ (free reduction).
- **Multiplication:** string concatenation, then reduce.
- **Identity:** the empty string ε.

Intuitively: $F(S)$ is the **most general** group you can build from S. That is, no relations other than the group axioms.

Formalizing finitely presented groups

From $F(S)$, add **relations**: pick a set $R \subseteq F(S)$ of words we want to declare equal to ε.

- Force $r = \varepsilon$ for all $r \in R$ by quotienting out the **normal closure** $\langle\langle R \rangle\rangle$, the smallest normal subgroup containing R.
  - (Need to quotient group by a normal subgroup to get back a group.)
- Result: $\langle S \mid R \rangle := F(S)/\langle\langle R \rangle\rangle$.
- Every element of R maps to the identity.
  - All other relations follow by normality and group axioms.

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.

FP groups sit at a crossroads between algebra, topology, geometry, logic, and computation. They are simple to define but rich enough to encode extremely complicated phenomena.

Many iterations

1.
```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (α : Type) (_: Finite α) (rels : Set (FreeGroup α)) (_ : rels.Finite),
  Nonempty (G ≃* (PresentedGroup rels))
```
LEAN

2.
```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out: ∃ S : Set G, S.Finite ∧ Subgroup.closure S = ⊤ ∧
  IsNormalClosureFG (FreeGroup.lift ((↑) : S → G)).ker
```
LEAN

3.
```
class IsFinitelyPresented (G : Type*) [Group G] : Prop where
  out : ∃ (n : ℕ) (f : FreeGroup (Fin n) →* G),
    Function.Surjective f ∧ IsNormalClosureFG f.ker
```
LEAN

✅ avoids `Type(*)`, `Nonempty`, ugly coersions, yet invariance proof flows.

(Optional) Lesson: Mathlib is about reusable API

- Here's what can happen when you use `Type*`.
  - While statements are correct, using `rw` on them causes metavar error.

```
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!
- #34624: surjection between types induces surjection between free groups
  - My original proof was using induction. (`d165011`)
  - Reviewer helped me break it down into little re-usable chunks. (`225e8a3`)
    - Each chunk is a "bridge" from A to B. **SUPER FUN DEMO OPEN ME.**

SECTION

You can contribute to mathlib!

Collaboration workflow

Collaborating is surprising smooth using Lean!

- Zulip to coordinate people.
- Github to PR request.
- Once definition is 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 a first-time contributor.
  - Would write filler PR. (Example.)

```lean
theorem my_old_statement := by
  sorry
```

  - 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** to help build A Dataset of Modern Formalized Theorem Statements with **Kevin Buzzard**'s group.

- If we provide Annals of Mathematics Lean statements to AI, how useful are they at actually autoformalizing frontier math?
- Main bottleneck are **definitions**.
- Want working mathematician's input.
  - Formalized math vs informal math defs differ, but want to get it right.
  - What is the minimal API needed to test that? What's important about def?
- Want assistance completing the API once definitions get merged.

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.
- Everyone at **MadLean** for sharing their knowledge with me.

Thank you!

- Thank you **Niki Vazou** for inviting me.
- Thank you **Daniel Jurjo Rivas** for 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] D. Zheng, I. von Glehn, Y. Zwols, and others, “AI co-mathematician: Accelerating mathematicians with agentic AI.” [Online]. Available: https://arxiv.org/abs/2605.06651

[9] M. Lackenby, “Every finite group admits a just finite presentation.” [Online]. Available: https://arxiv.org/abs/2605.10402

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

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

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

The end!