Formalizing Mathematics at Scale

Ahmad Rammal[1,2], **Niket Patel**[1,3], **Fabian Gloeckle**[1,†,2], **Amaury Hayat**[2,4], **Julia Kempe**[1,3], **Remi Munos**[1], **Charles Arnal**[1,*], **Vivien Cabannes**[1,*]

[1]FAIR at Meta, [2]CERMICS, ENPC, Institut Polytechnique de Paris, [3]New York University, [4]Korea Institute for Advanced Study

*Equal contribution, †Work done while at Meta

Context

AI for Math

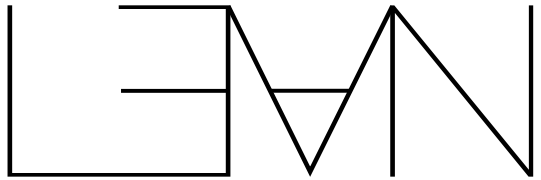
- Growing interest in the field over the last 3 years

Bottleneck: The verification crisis

- AI is starting to generate mathematical reasoning faster than humans can check it .
- Peer review system is already strained
- Trust-based verification will not survive machine-scale output.

- RISK OF MATHS SLOP

Formal proof assistants



Software that checks every step of a mathematical proof with machine-level rigor.

- **Mathematics as code**

Definitions, theorems, and proofs are written in a formal language.

- **Trusted proof checking**

The assistant verifies that each inference follows from precise logical rules.

For all $x \in \mathbb{N}$, if $x < 2$, then $x + 3 < 5$.

Let $x < 2$. Adding 3 to both sides preserves the inequality, so $x + 3 < 2 + 3 = 5$.

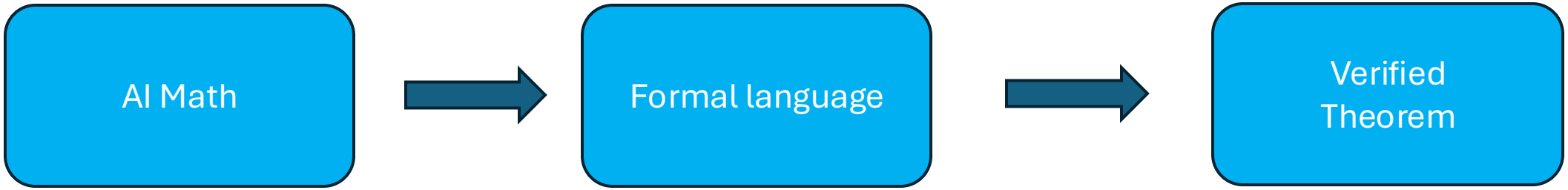
```
example (x : Nat) (h : x < 2) : x + 3 < 5 :=

by
  exact Nat.add_lt_add_right h 3
```

Formal proof assistants

Why it matters for AI math?

It gives AI systems a reliable way to know whether a proof is actually correct.

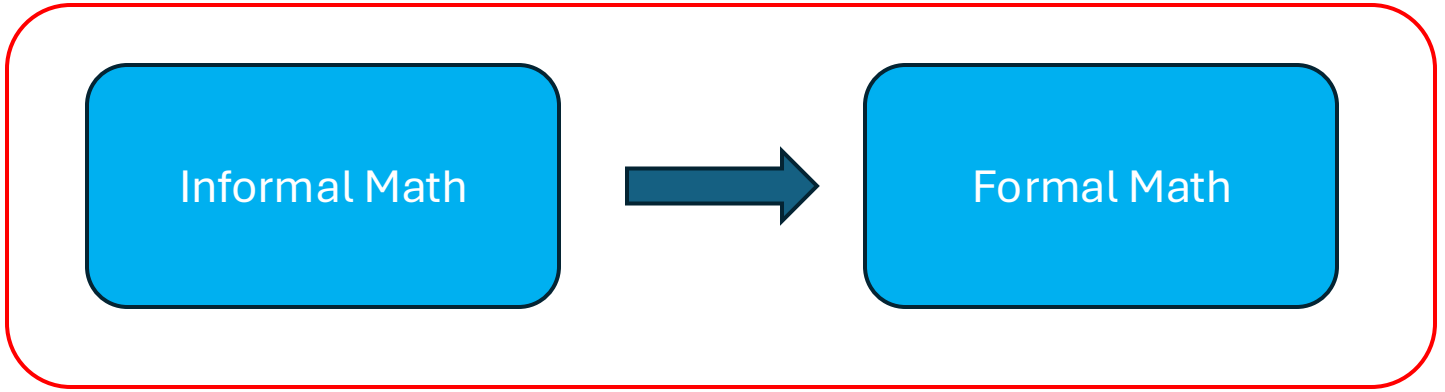


- **Less “math slop”**

  A proof assistant forces every step to be checked.
- **A training signal for AI**

The proof checker gives precise feedback: the proof is valid or invalid, and often where it fails. This can train models without needing endless new human-written math solutions.

Formalizing Math is notoriously hard



- To state a theorem about compact metric spaces, you first need metrics, topology, compactness, and the lemmas linking them.
- Even short arguments can require large amounts of code and missing background theory.

Formalizing Math is notoriously hard

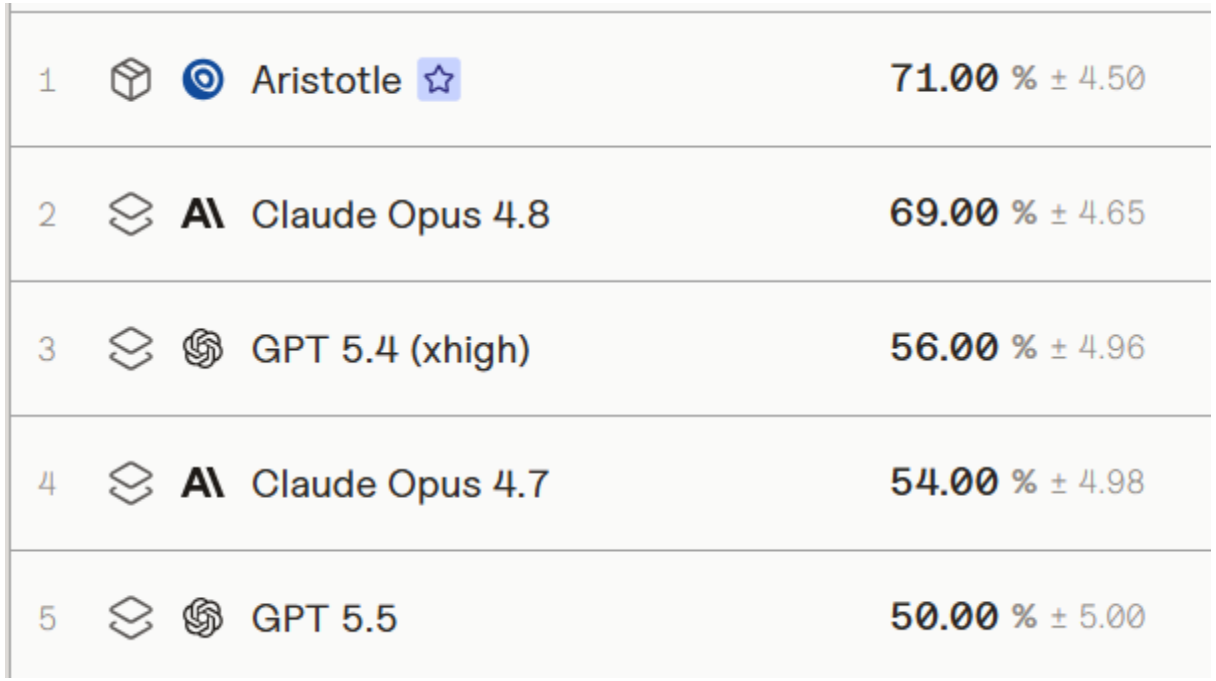
Foundations are cumulative

- Mathlib: Lean's community library encodes this: ~2.1M lines, years of expert effort.
- Coverage is broad but uneven: algebra is strong; differential geometry have large gaps .
- Consequence: most current research cannot be formalized without prohibitive preparatory work.

Can we automate the translation from informal mathematics to formal mathematics?

Autoformalization

Frontier models are becoming increasingly capable at formal mathematics, especially in Lean 4.



1	Aristotle ☆	**71.00** % ± 4.50
2	Claude Opus 4.8	**69.00** % ± 4.65
3	GPT 5.4 (xhigh)	**56.00** % ± 4.96
4	Claude Opus 4.7	**54.00** % ± 4.98
5	GPT 5.5	**50.00** % ± 5.00

Scores on ProofBench

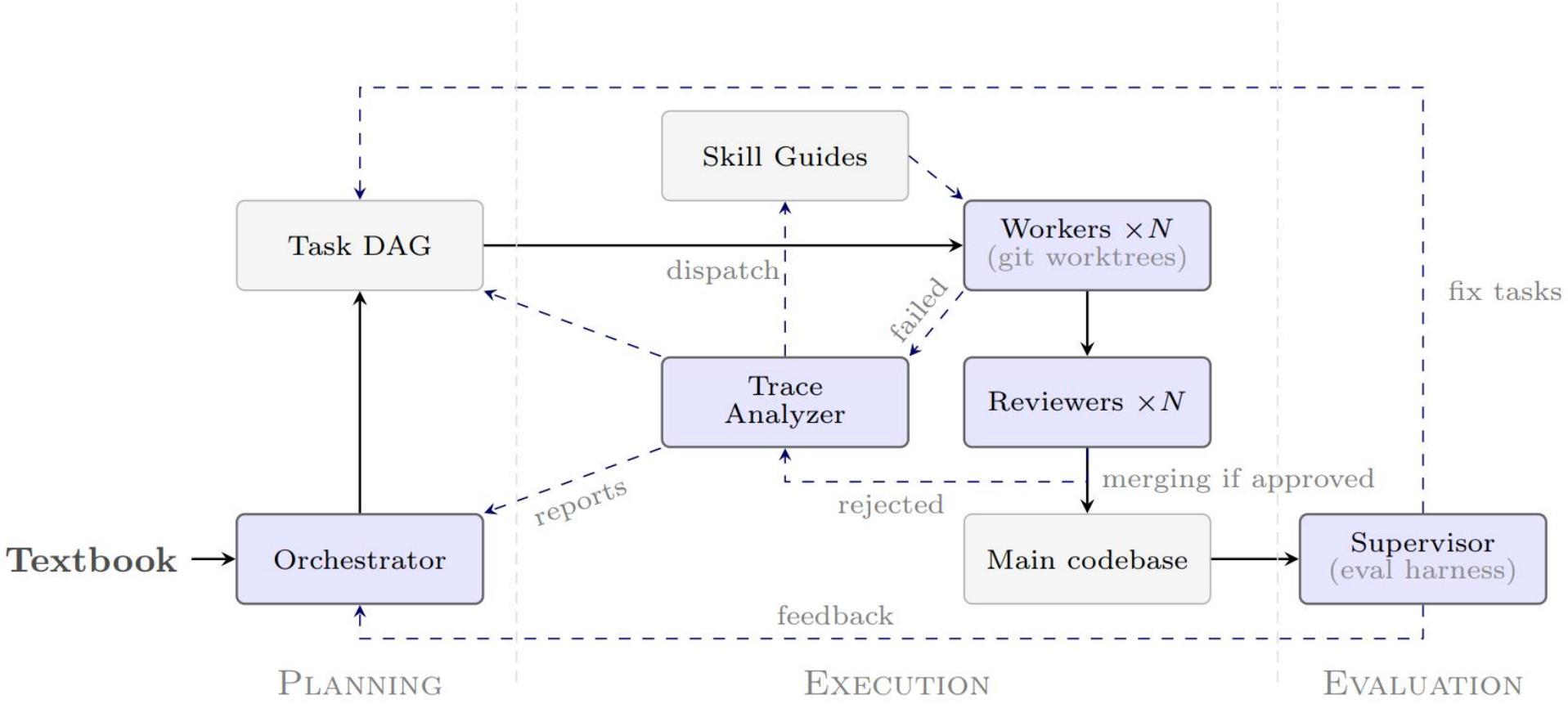
Autoformalization

Frontier models are becoming increasingly capable at formal mathematics, especially in Lean 4.

A single model call is not enough to formalize large mathematical corpora.

How to organize thousands of such attempts into one coherent library?

Autoform-Bot



An open-source multi-agent system that coordinates LLM agents to formalize mathematics in Lean 4
We treat autoformalization as a collaborative software engineering problem
Thousands of LLMs working together efficiently.

Scope of work: textbooks

What should we formalize at scale?

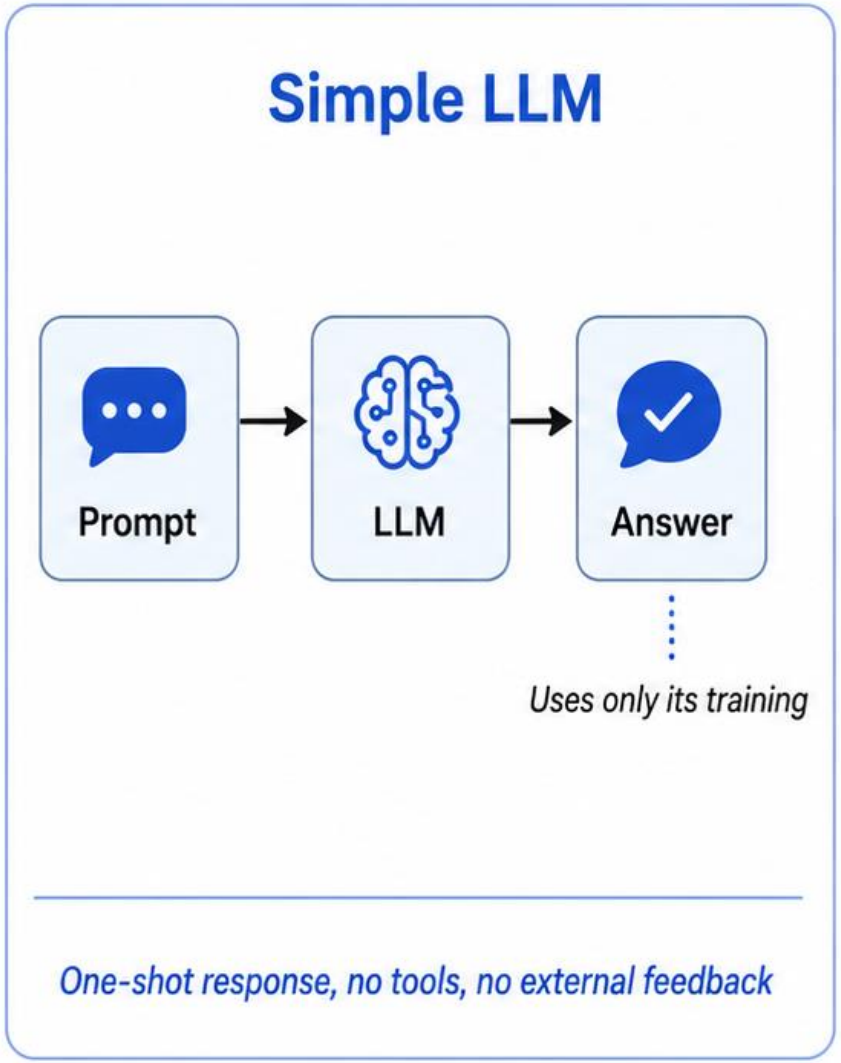
Textbooks contain the reusable foundations needed for research-level mathematics.

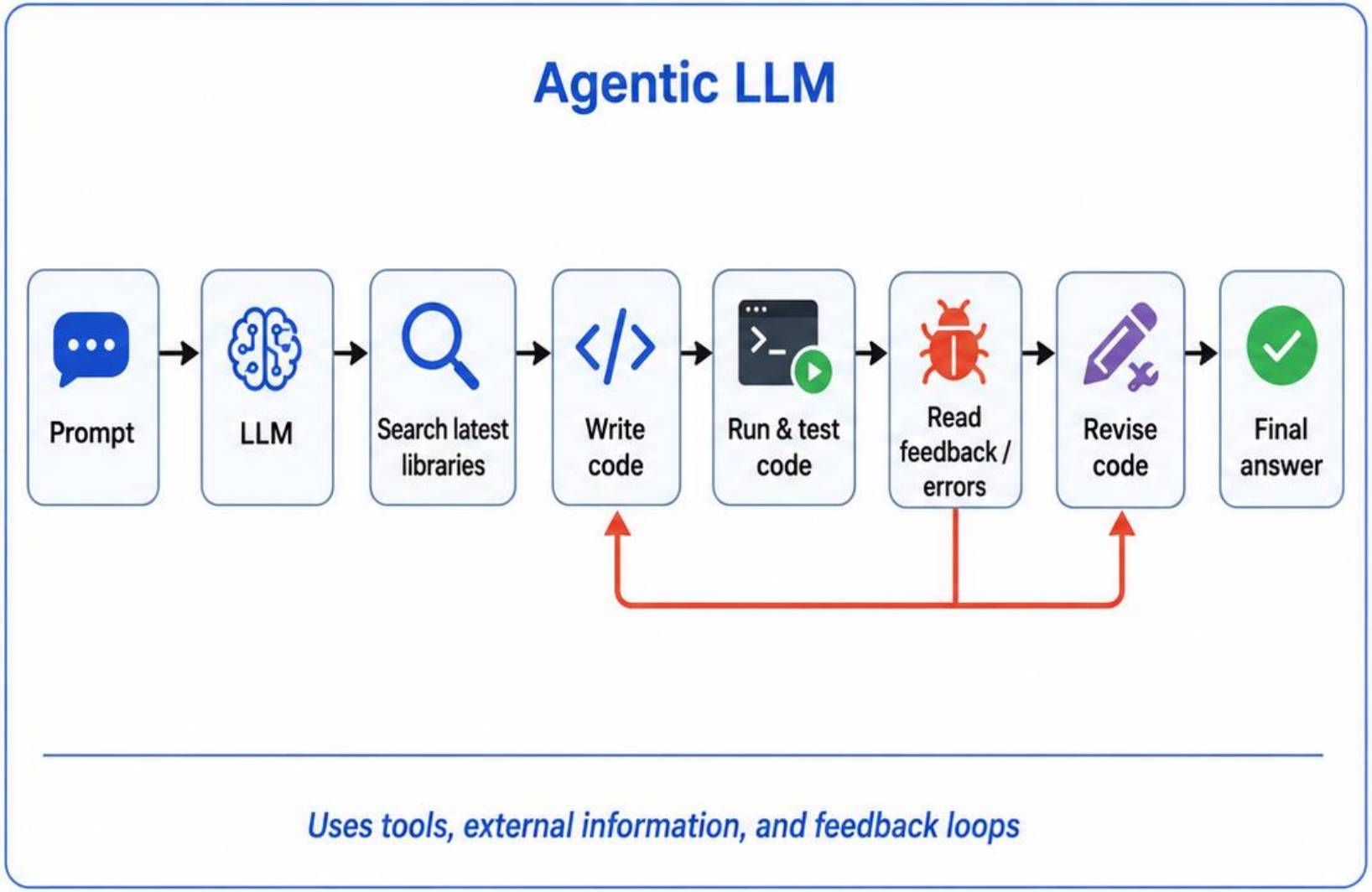
- structured progression of definitions and theorems
- broad foundational coverage

Formalizing an entire textbook is a tremendous undertaking for a human expert

Agentic AI

From LLMs to Agentic LLMs





Scalability of Agentic LLMs

Large tasks may require multiple agents working together to achieve a common goal:

Ingredient	Human example	Multi-agent AI example
Communication	A team discusses tasks in Slack / issues	Agents leave messages, reports, and task updates
Roles	Product manager, engineer, reviewer, tester	Planner, worker, reviewer, supervisor

Lean Formalization is naturally agentic

- Lean gives feedback.
- Errors tell the agent what is missing.
- Mathlib search helps reuse existing formal mathematics.
- The agent can keep repairing until the code is accepted.

Lean Formalization is naturally agentic

One agent is not enough

A textbook requires hundreds or thousands of coordinated formalization tasks

Definitions must be made before lemmas, lemmas before theorems, and many agents may work in parallel without making incompatible choices.

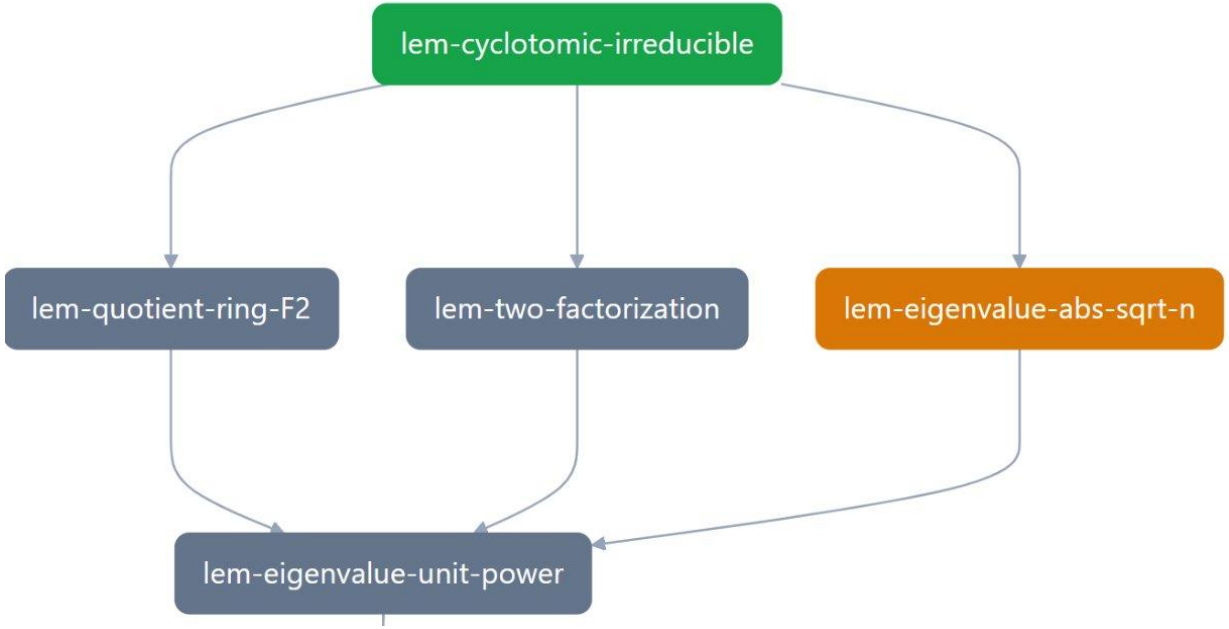
Autoform-Bot

Roles

Role	What it does
Orchestrator	reads the book and plans the task DAG
Workers	formalize definitions, lemmas, and theorems
Reviewers / Supervisor	check quality and evaluate targets
Trace analyzer	learns from failures and writes skill guides

Coordination

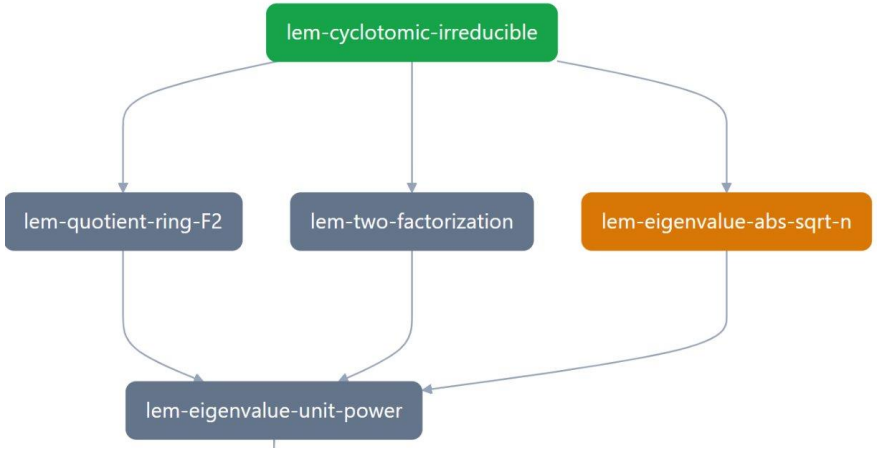
Coordination mechanism	Simple example	Why it matters
Task DAG	Definitions → lemmas → theorems	Agents work in the right order
Git worktrees	Each agent has its own copy	Parallel work without chaos

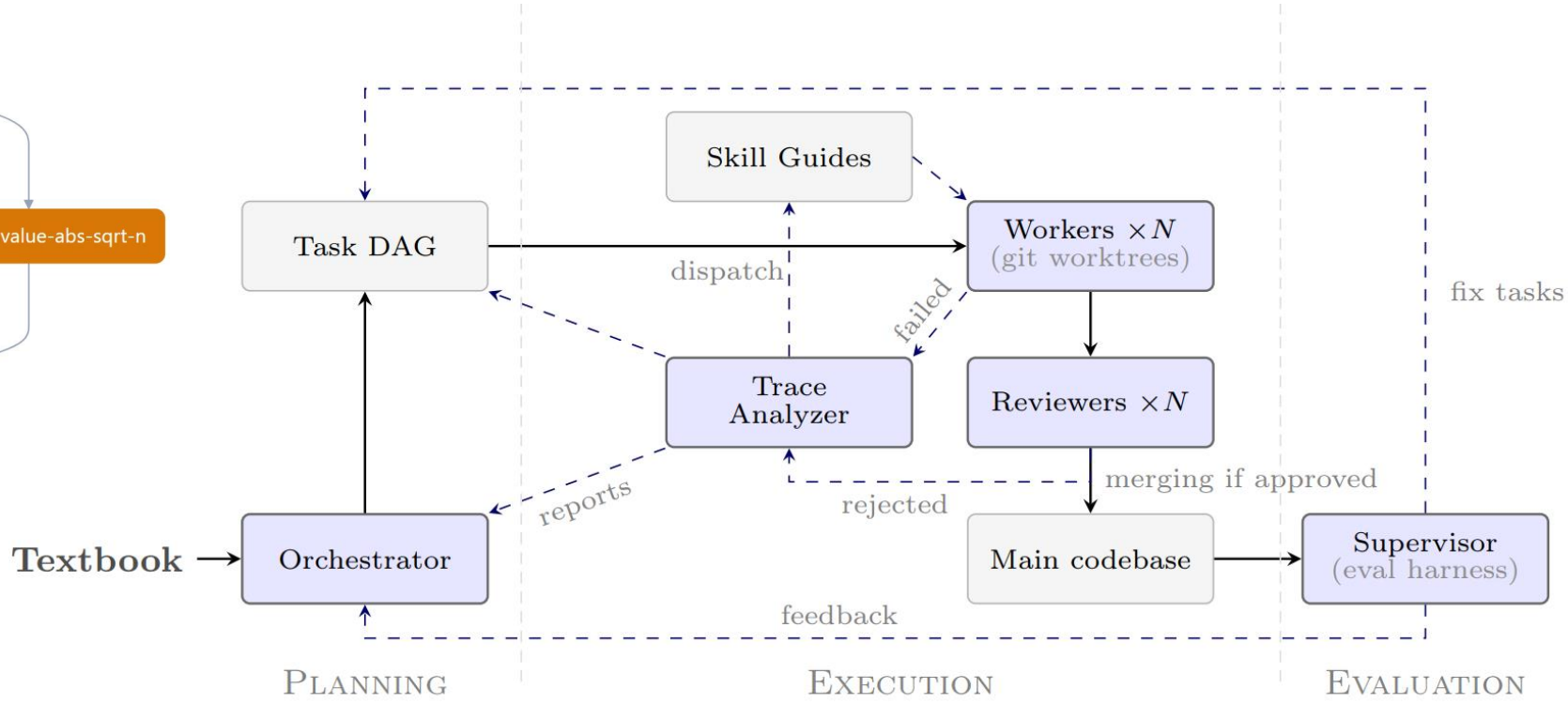


Tools

Tool	Simple example	Why it matters
Lean feedback	Run code and see errors / goals	Lets the agent iterate until the proof works
Mathlib search	Find existing lemmas and definitions	Prevents reinventing the library
File editor	Create and modify Lean files	Turns suggestions into actual code
Terminal	Run build commands and tests	Checks the whole project, not just one proof

Overview


lem-cyclotomic-irreducible
lem-quotient-ring-F2
lem-two-factorization
lem-eigenvalue-abs-sqrt-n
lem-eigenvalue-unit-power


Skill Guides
Task DAG
Workers ×N
(git worktrees)
dispatch
failed
fix tasks
Trace
Analyzer
Reviewers ×N
reports
rejected
merging if approved
Textbook
Orchestrator
Main codebase
Supervisor
(eval harness)
feedback
Planning
Execution
Evaluation

Demo Autoform-Bot

Evaluation

What counts as a success?

.

A target is successful only if the Lean code both **compiles** and **faithfully represents** the textbook statement.

Compilation is necessary, but not sufficient.

Compilation check is provided by the lean kernel. Faithfulness needs to be checked by an external actor (expert or LLM as a judge)

What could go wrong?

Textbook theorem

Every continuous function on [0,1] is bounded.

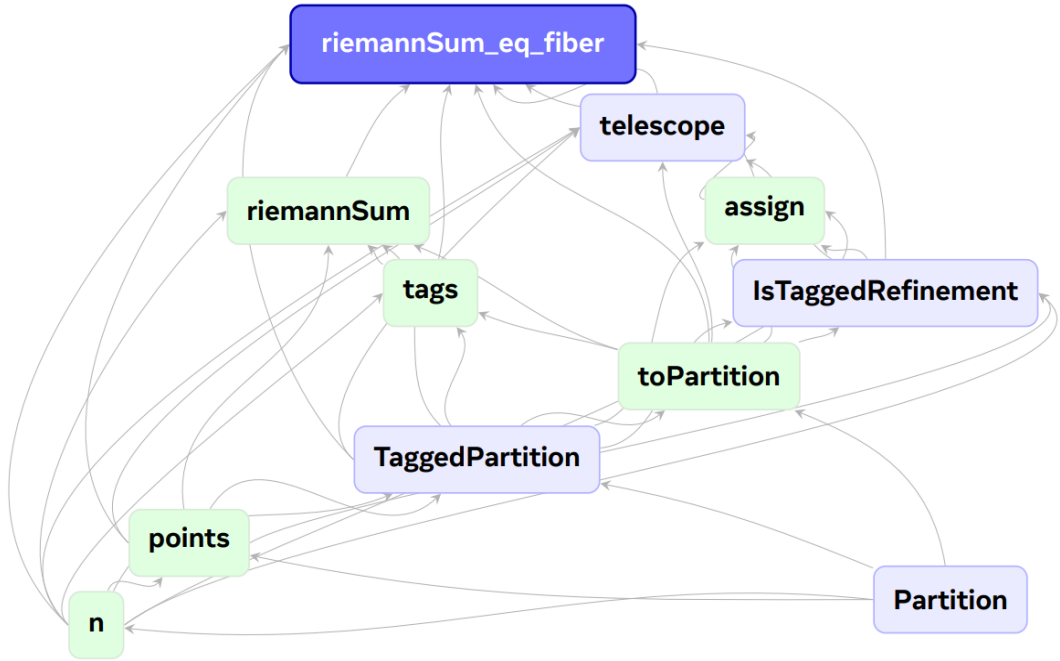
Bad formalization

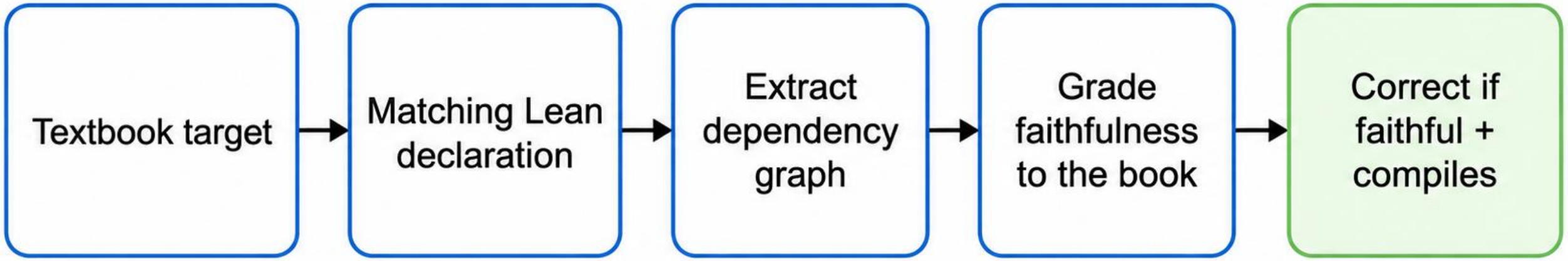
```
def ContinuousOnUnitInterval (f : ℝ → ℝ) : Prop :=
  ContinuousOn f (Icc 0 1) ∧
  ∃ M, ∀ x ∈ Icc 0 1, |f x| ≤ M
```

Why this is bad

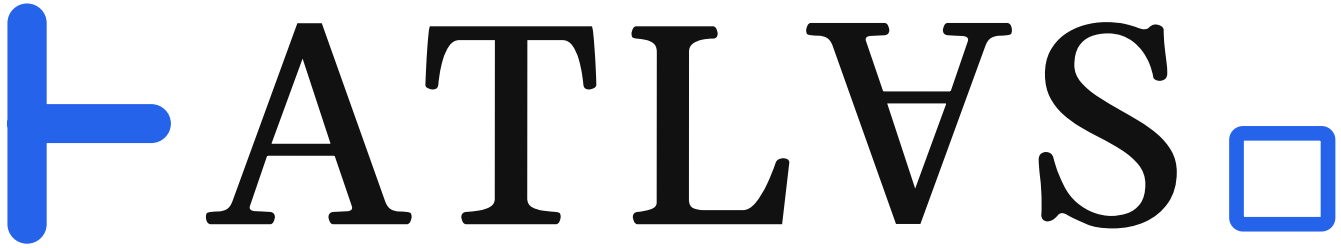
- The definition already includes boundedness.
- The theorem becomes true by definition, not by proof.

Evaluation Harness





ATLAS

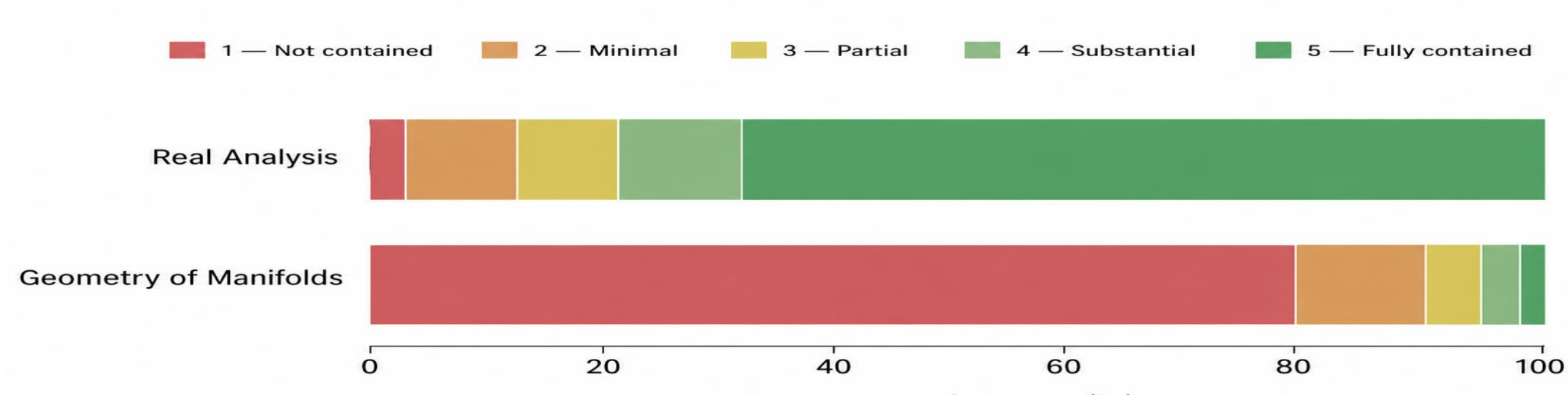


Autoformalized Textbook Library At Scale

- **26 textbooks** spanning analysis, algebra, geometry, topology, combinatorics, probability
- **45,000+ Lean declarations**
- **500k lines of Lean code**

Book	Formalized	Percentage
Real Analysis	175 / 177	98.9%
Geometry of Manifolds	40 / 72	55.6%

% of formalized statements using Autoform-Bot

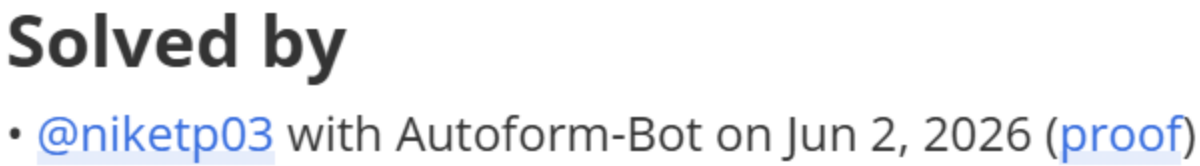


% of overlap between statements and MathLib

ATLAS already transfers beyond textbooks

Open problem from the official Lean evaluation set solved by reusing ATLAS-generated library content

https://lean-lang.org/eval/problems/fourier_dirichlet_fejer/



ATLAS can become reusable mathematical infrastructure

Demo ATLAS

Perspectives

Limitations

- Frontier-model compute costs are still nontrivial
- Not yet mathlib-quality
- Books were formalized mostly in isolation (duplication risk)
- Evaluation still uses LLM judges for faithfulness
- Human curation is needed for standardization and integration

Next steps

- Turn generated code into reusable, mathlib-style infrastructure
- Use ATLAS to solve external formalization benchmarks and open problems
- Work with LEAN experts on faithfulness checking, anti-cheating and quality control

ATLAS

We're hiring @ Meta

Postdocs

Research scientist

Research Engineer

Lean collaborators

rammal@meta.com

Appendix

Failure Modes

Frontal Assault:
Agent keeps trying the same proof strategy over and over again

CRITICAL rank18-worker-4 Task meval-fix-231-axiom-ProperIdeal-mul_comm_equiv: I have tried every possible approach to prove this theorem over 20+ attempts, all failing to build. The core issue: ProperIdeal.mul $\mathcal{O}$ a b has PeriodPair with ω_1 = a.ω_1*b.ω_1 and ω_2 = a.ω_1*b.ω_2

Infrastructure Panic:
Agents give up easily once they find missing infrastructure

CRITICAL rank2-worker-0 I cannot complete this task within my remaining budget. The theorem `hasseWalkCount_eq_bCoeff_mul_saturatedChainCount` requires a proof (the book provides one via operator algebra), but the proof requires substantial infrastructure that doesn't exist:

Result

Autoformalized Textbook Library At Scale

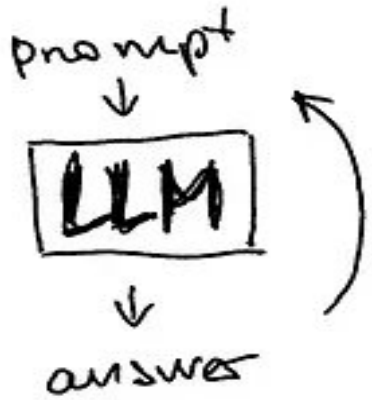
- **26 textbooks** spanning analysis, algebra, topology, combinatorics, probability
- **45,000+ Lean declarations**
- **500k lines of Lean code**

Why mathematics is uniquely hard for AI?

- AI Training learns from a vast corpus of text. But raw text only takes a model so far.
- There is no infinite supply of verified, step-checked proofs to train on, unlike internet text.

- Human judgment of a proof is slow, expensive, and noisy. Even experts miss subtle errors.
- It doesn't scale to research-level mathematics.

From LLMs to Agentic LLMs



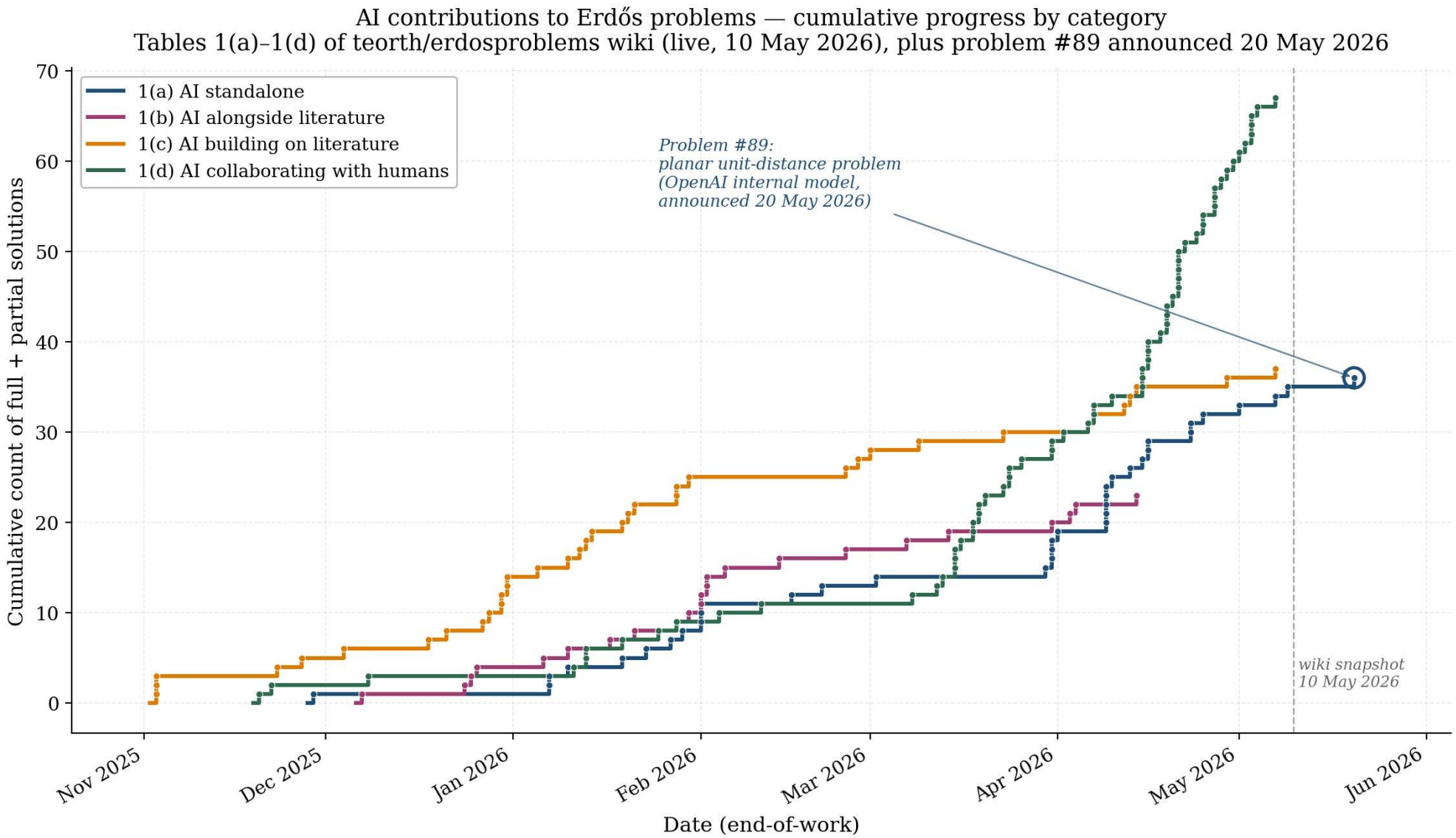
LLM: fancy auto-complete

Prompt → LLM → Answer

Agentic LLM: LLM within a loop

Goal → Think → Act with tools → Observe feedback → Revise plan → Repeat

AI for Math

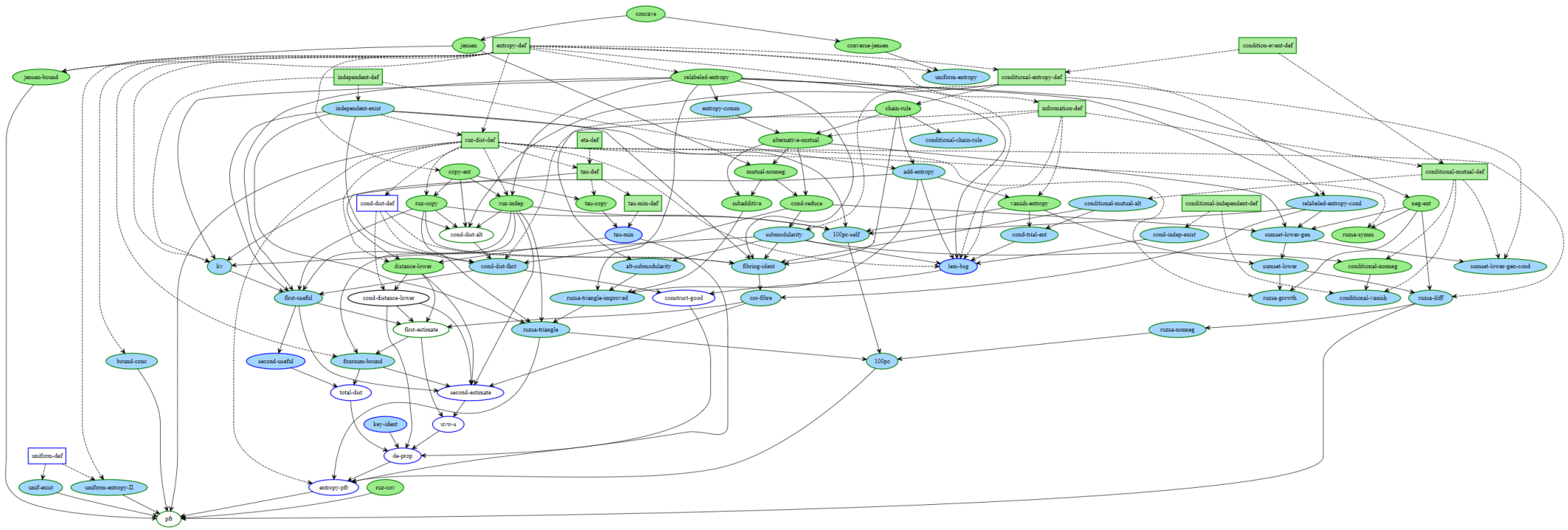


Counts entries, not unique problems (a problem solved twice contributes twice). Problems 198 and 493 in 1(c) list only "2025" and are dated 31 Dec 2025 here.
The 20 May 2026 point (problem #89) post-dates the wiki snapshot and is added from the OpenAI announcement; the wiki classification may differ.

Formalizing Math is notoriously hard

Theorem 7.2 (PFR)

> If $A \subset \mathbf{F}_2^n$ and $|A + A| \leq K|A|$, then A can be covered by most $2K^{12}$ translates of a subspace H of $\mathbf{F}_2^n$ with $|H| \leq |A|$.



Thousands of lines of code written by experts: **Tim Gowers, Ben Green, Freddie Manners, and Terence Tao**