Agentic approach for neural theorem proving in Rocq

CANUM 2026

Guillaume Baudart, Emilio Jesús Gallego Arias, Marc Lelarge, Théo Stoskopf, et Jules Viennot

The outbreak of reasoning models

P. Erdö ce Prob lem

https:/

MATHEMATICAL EXPLORATIO

BOGDAN GEORGIEV, JAVIER GÓMEZ-SERRANO

ABSTRACT. AlphaEvolve, introduced in [224], is a gener
capabilities of LLMs with automated evaluation in an iterati
algorithmic solutions to challenging scientific and practica
a tool for autonomously discovering novel mathematical
standing open problems.
To demonstrate its breadth, we considered a list of 67
geometry, and number theory. The system rediscovered the
improved solutions in several. In some instances, AlphaEv
of input values into a formula valid for all input values.
with Deep Think [149] and AlphaProof [148] in a broa
reasoning systems provide automated proof generation and
These results demonstrate that large language model-gui
ematical constructions that complement human intuition, at
highlighting the potential for significant new ways of interac
AlphaEvolve as a powerful tool for mathematical discover
optimization problems at scale, often with significantly red

1. INTRO

The landscape of mathematical discovery has been func
tional tools that can autonomously explore mathematical
291]. AlphaEvolve (see [224]) represents a step in thi
when combined with evolutionary computation and rig
structions that either match or improve upon the best-kno
large scales.

AlphaEvolve is not a general-purpose solver for all typ
to attack problems in which a key objective is to const
quantitative properties, such as obeying a certain inequ
up paper, we report on our experiments testing the per
problems, primarily in the areas of analysis, combinat
provided by AlphaEvolve were not merely numerical in
mathematicians, by other tools such as Deep Think, an
able to match or exceed previous results in all cases, a
achieve could likely also have been matched by more trad
by human experts. However, in contrast to such methods,
up to study large classes of problems at a time, without requiring extensive expert supervision for
problem. This demonstrates that evolutionary computational approaches can systematically explore th
mathematical objects in ways that complement traditional techniques, thus helping answer questions
relationship between computational search and mathematical existence proofs.

We have also seen that in many cases, besides the scaling, in order to get AlphaEvolve to output co
results to the literature and in contrast to traditional ways of doing mathematics, very little overhead

The authors are listed in alphabetical order.

1

THE MOTIVIC CLASS OF THE SPACE OF GENUS 0 MAPS TO THE FLAG VARIETY

JIM BRYAN, BALÁZS ELEK, FREDDIE MANNERS, GEORGE SALAFATINOS, AND RAVI VAKIL

ABSTRACT. Let Fl_{n+1} be the variety of complete flags in $\mathbb{A}^{n+1}$ and let $\Omega^2_\beta(\mathrm{Fl}_{n+1})$ be the space of based maps $f : \mathbb{P}^1 \to \mathrm{Fl}_{n+1}$ in the class $f_*[\mathbb{P}^1] = \beta$. We show that under a mild positivity condition on β, the class of $\Omega^2_\beta(\mathrm{Fl}_{n+1})$ in $K_0(\mathrm{Var})$, the Grothendieck group of varieties, is given by

$$[\Omega^2_\beta(\mathrm{Fl}_{n+1})] = [\mathrm{GL}_n \times \mathbb{A}^a].$$

The proof of this result was obtained in conjunction with Google Gemini and related tools. We briefly discuss this research interaction, which may be of independent interest. However, the treatment in this paper is entirely human-authored (aside from excerpts in an appendix which are clearly marked as such).

1. INTRODUCTION

Let $\mathrm{Fl}_{n+1} = \mathrm{GL}_{n+1}/B$ be the complete flag variety which parameterizes flags of quotients:

$$\Bbbk^{n+1} \twoheadrightarrow E_n \twoheadrightarrow \cdots \twoheadrightarrow E_1$$

where $\dim E_k = k$. There is a universal sequence of vector bundle quotients on Fl_{n+1}:

$$\mathcal{O}^{\oplus n+1}_{\mathrm{Fl}_{n+1}} = \mathcal{E}_{n+1} \twoheadrightarrow \mathcal{E}_n \twoheadrightarrow \cdots \twoheadrightarrow \mathcal{E}_1.$$

Let $\Omega^2_\beta(\mathrm{Fl}_{n+1})$ be the space of genus zero based maps in the class β:

$$\Omega^2_\beta(\mathrm{Fl}_{n+1}) = \left\{ f : \mathbb{P}^1 \to \mathrm{Fl}_{n+1} : \quad f_*([\mathbb{P}^1]) = \beta, \quad f([1:0]) = [\mathrm{Id}] \right\}$$

where $\beta \in A_1(\mathrm{Fl}_{n+1})$ and $[\mathrm{Id}] \in \mathrm{GL}_{n+1}/B$ is the base point in Fl_{n+1}.

There is an isomorphism $A_1(\mathrm{Fl}_{n+1}) \cong \mathbb{Z}^n$ such that the n-tuple associated to β is given by $(d_1, \ldots, d_n)$ where

$$d_i = \deg(f^* \mathcal{E}_i).$$

We say that $\beta = (d_1, \ldots, d_n)$ is *strictly monotonic* if $0 < d_n < d_{n-1} < \cdots < d_1$. The main result of this paper is the following:

Theorem 1.1. *Suppose $\beta = (d_1, \ldots, d_n)$ is strictly monotonic. Then we have the following equality in $K_0(\mathrm{Var}_{\Bbbk})$, the Grothendieck group of varieties over $\Bbbk$:*

$$[\Omega^2_{d_n,\ldots,d_1}(\mathrm{Fl}_{n+1})] = \left[\mathrm{GL}_n \times \mathbb{A}^{D-n^2}\right]$$

where $D = \sum_{k=1}^n 2d_k$.

In particular, Theorem 1.1 gives an equality of point counts over finite fields (see Corollary A.1).

Our original interest in the variety $\Omega^2_\beta(\mathrm{Fl}_{n+1})$ is as an algebraic counterpart to the double loop space in topology:

$$\Omega^2_{\beta,\mathrm{top}}(\mathrm{Fl}_{n+1}) = \left\{ f : S^2 \to \mathrm{Fl}_{n+1},\ f_*([S^2]) = \beta, f([1:0]) = [\mathrm{Id}] \right\}$$

1

with $x_0 \in \mathbb{R}^n$. Gradient descent is known to conver
$\mathcal{O}(1/k)$ and

$$x_k \to x_\infty \in \arg\min$$

[27, 9, 14]. However, the $\mathcal{O}(1/k)$ rate is famously su
Nesterov's 1983 seminal paper [24] presented the Nes
method

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k),$$
$$y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k)$$

with $x_0 = y_0 \in \mathbb{R}^n$, $t_0 = 1$, and $t^2_{k+1} - t_{k+1} \le t^2_k$.
sense of $f(x_k) - \inf f \le \mathcal{O}(1/k^2)$, an accelerated rat
or $t_k = (k+2)/2$ for $k = 0, 1, \ldots$. However, whether

$$x_k \overset{?}{\to} x_\infty \in \arg\min$$

has remained an open problem [17, 4, 16, 10].

In this work, we resolve this longstanding ope
showing point convergence for NAG in the sense of

$$x_k \to x_\infty, \qquad y_k \to x_\infty, \qquad x_$$

The discovery of the proof was heavily assisted b
language model, and we describe the process throug

*Submitted to the editors DATE.
†Department of Mathematics, University of California, Los Angeles (uijeongjang@math.ucla.edu, eryu@math.ucla.edu).

1

Google DeepMind 2026-5-22

Advancing Mathematics Research with AI-Driven Formal Proof Search

George Tsoukalas[1†], Anton Kovsharov[1†], Sergey Shirobokov[1†], Anja Surina[1], Moritz Firsching[1], Gergely Bérczi[2], Francisco J. R. Ruiz[1], Arun Suggala[1], Adam Zsolt Wagner[1], Eric Wieser[1], Lei Yu[1], Aja Huang[1], Miklós Z. Horváth[1], Andrew Ferraiuolo[1], Henryk Michalewski[1], Codrut Grosu[3], Thomas Hubert[1], Matej Balog[1], Pushmeet Kohli[1] and Swarat Chaudhuri[1†]
[1]Google DeepMind, [2]Aarhus University, [3]Google

Large language models (LLMs) increasingly excel at mathematical reasoning, but their unreliability limits their utility in mathematics research. A mitigation is using LLMs to generate formal proofs in languages like Lean. We perform the first large-scale evaluation of this method's ability to solve open problems. Our most capable agent autonomously resolved 9 of 353 open Erdős problems at the per-problem cost of a few hundred dollars, proved 44/492 OEIS conjectures, and is being deployed in combinatorics, optimization, graph theory, algebraic geometry, and quantum optics research. A basic agent alternating LLM-based generation with Lean-based verification replicated the Erdős successes but proved costlier on the hardest problems. These findings demonstrate the power of AI-aided formal proof search and shed light on the agent designs that enable it.

1. Introduction

Large language models (LLMs) have recently shown remarkable promise in solving complex mathematics problems [21, 65], but *unreliability* remains a primary barrier to their integration into mathematics research. Because LLM-generated natural language proofs can contain subtle logical errors, or "hallucinations," they require expensive expert review. Mistakes in unreviewed intermediate steps can cascade through a proof, limiting the complexity of tasks that can be delegated to AI.

Recent efforts [29, 1] mitigate these issues by using AI to generate proofs in formal languages like Lean [43], in which a compiler automatically verifies every logical step. So far, successes of this paradigm have been concentrated in competition mathematics and the human-aided formalization of natural language arguments [28]. In this paper, we demonstrate its broader potential through a large-scale evaluation on open research-level problems.

To this end, we developed a framework, AlphaProof Nexus, for LLM-aided proof generation and used it to build a basic agent in which a set of subagents independently searches for proofs with feedback from the Lean compiler. We also developed a "full-featured" agent in which subagents are coordinated using an evolutionary algorithm [46] and can use AlphaProof [29], a system for olympiad-level Lean theorem-proving based on reinforcement learning, as a focused proof tool.

Our full-featured agent autonomously solved 9 Erdős problems out of 353 attempted, including two questions that had been open for 56 years [54, 7, 17], at the inference cost

[1†]Equal contributions. The first three authors are in random order. Correspondence to pushmeet@google.com and swarat@google.com.

EARLY EVIDENCE OF VIBE-PROVING WITH CONSUMER LLMS: A CASE STUDY ON SPECTRAL REGION CHARACTERIZATION WITH CHATGPT-5.2 (THINKING)

Brecht Verbeken[1,2,§] 0000-0002-7506-3298
Brando Vagenende[1] 0000-0002-0573-8093
Marie-Anne Guerry[1] 0000-0001-5842-8905
Andres Algaba[1,2] 0000-0002-0532-3066
Vincent Ginis[1,2,3] 0000-0003-0063-9608
[1]Data Analytics Lab, Vrije Universiteit Brussel, Pleinlaan 5, 1050 Brussel, Belgium
[2]imec-SMIT, Vrije Universiteit Brussel, Pleinlaan 9, 1050 Brussels, Belgium
ol of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts 02138, USA

February 24, 2026

ABSTRACT

rge Language Models (LLMs) are increasingly used as scientific copilots, but evidence on their le in research-level mathematics remains limited, especially for workflows accessible to individual searchers. We present early evidence for vibe-proving with a consumer subscription LLM through auditable case study that resolves Conjecture 20 of Ran and Teng (2024) on the exact nonreal ectral region of a 4-cycle row-stochastic nonnegative matrix family. We analyze seven shareable atGPT-5.2 (Thinking) threads and four versioned proof drafts, documenting an iterative pipeline generate, referee, and repair. The model is most useful for high-level proof search, while human perts remain essential for correctness-critical closure. The final theorem provides necessary and fficient region conditions and explicit boundary attainment constructions. Beyond the mathematical ult, we contribute a process-level characterization of where LLM assistance materially helps d where verification bottlenecks persist, with implications for evaluation of AI-assisted research rkflows and for designing human-in-the-loop theorem proving systems.

AI-assisted mathematics · Karpelevich region · large language models · stochastic matrices

duction

, where programmers steer Large Language Models (LLMs) to generate software through high-level natural- tent rather than line-by-line specification, has rapidly transformed code generation [1, 2, 3]. Underpinning the sustained capability growth of frontier LLMs across successive systems and evaluations [4, 5, 6, 7]. odels grow more capable, they are increasingly deployed as scientific collaborators: generating candidate as in controlled studies [8, 9], acting as agentic research assistants that plan, search, and iterate over literature ents [10, 11, 12, 13, 14, 15, 16], and accelerating evidence synthesis and literature screening [17, 18, 19]. le, LLM tooling is associated with measurable shifts in scientific production and communication [20], ins unclear how reliably these systems support auditable, research-level mathematical workflows under ccess conditions [21, 22].

ponding author: brecht.verbeken@vub.be

1

helicity particles and $n-2$ plus-helicity gluons, which for generic (complexified) kinematics at tree level is the maximally allowed number [4, 11–14]. This gives them a privileged role in the theory, enabling their use as efficient building blocks for the full Yang–Mills theory.

In general, $n-2$ is actually *not* the maximally allowed number of plus gluons. In this paper, we show that $n-1$ plus (or "single-minus") amplitudes are in fact allowed even at tree level[2] with restricted "half-

[1] The aforementioned agreement between theory and experiment required over a half century of analytic and numerical work.
[2] Witten [5] notes that single-minus tree amplitudes are supported at a point in twistor space; see also [6]. All-plus and single-minus amplitudes are generically corrected at loop level [15, 16].

Mills theory (SDYM) [18], a restricted sector of Yang–Mills, and potentially resolve a puzzle therein. In general, the tree amplitudes of the Feynman expansion are thought to be equivalent to the fully nonlinear classical theory. However, on the one hand the classical solution space of SDYM is extremely nontrivial [19–21], while the tree diagrams were previously supposed to yield trivial two-point and three-point expressions. The latter seem insufficient to reproduce the former. Potentially, the single-minus tree amplitudes in SDYM found here resolve this tension.

[3] The half-collinear condition can be viewed as restricting the ingoing and outgoing momenta to a one-dimensional null circle on the celestial torus at the boundary of Klein space.

Are humans still useful?

YES

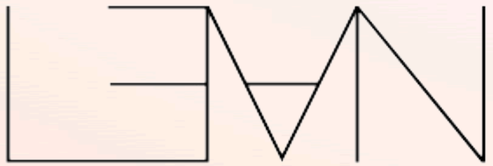
- To **guide** Large Language Models (LLMs)
  - by hinting
  - by discussing
  - by building environments

- To **review** the generated proofs ⟶ Can it be automated?

ROCQ

A trustworthy, industrial-strength interactive theorem prover and dependently-typed programming language for mechanised reasoning in mathematics, computer science and more.

Install | About The Rocq Prover | Why Rocq?



Latest Rocq Prover release: 8.20.1

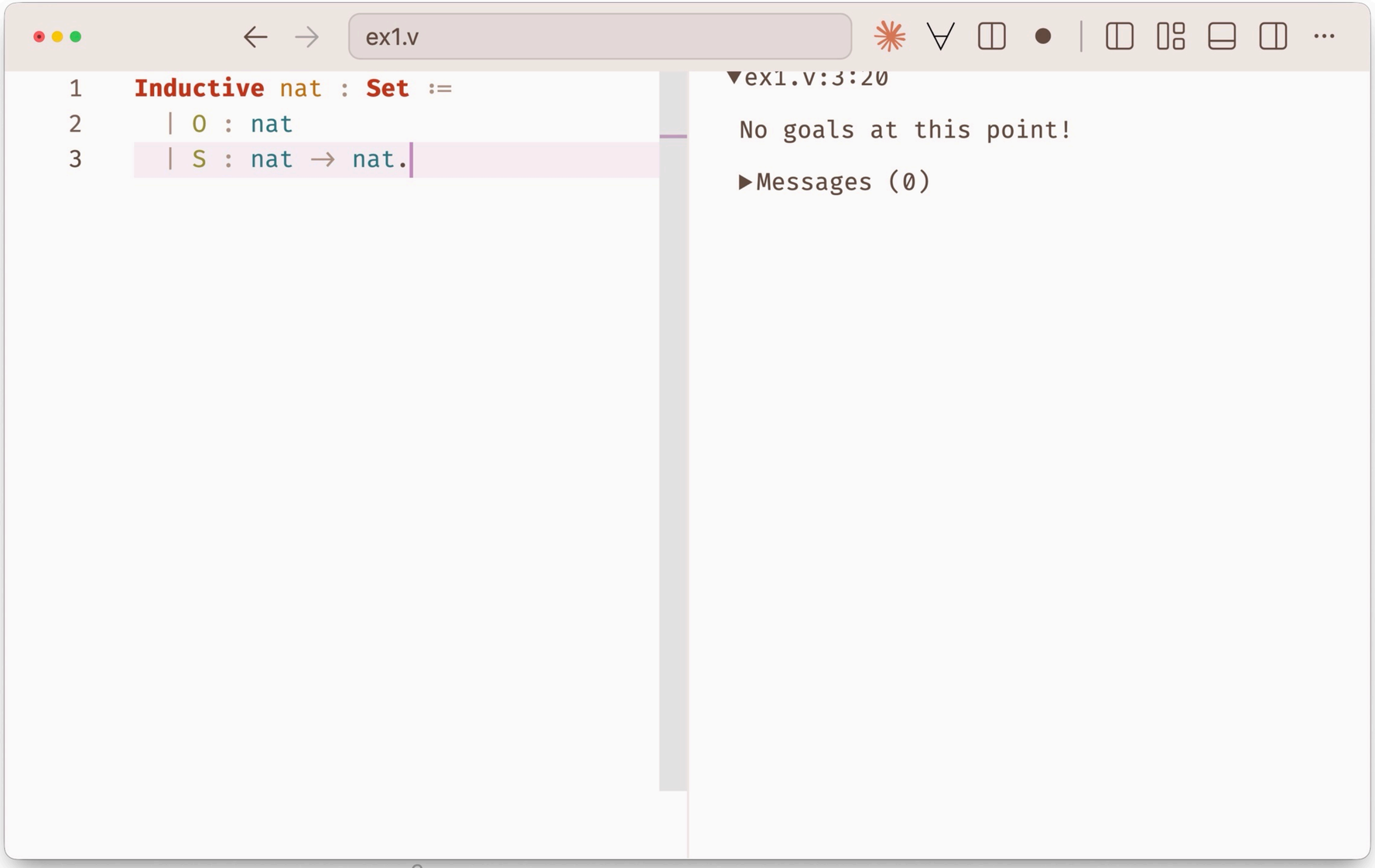
Latest Rocq Prover release candidate: 9.0+rc1

Latest Rocq Platform release: 2024.10.1

The Rocq Prover was formerly known as the Coq Proof Assistant (see more on **the name evolution**).

Formalizing mathematics

Write definitions:
Peano natural numbers

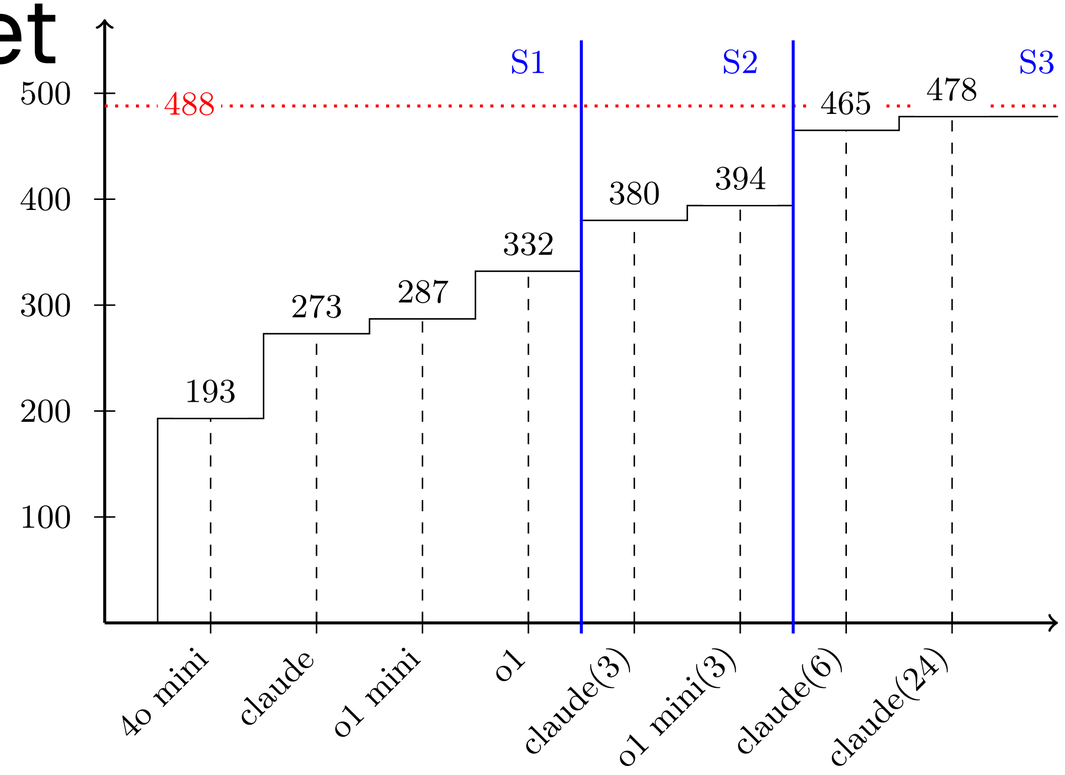


Specialized models

Can general models write in Rocq?

MiniF2F dataset

Category	Subcategory
IMO	
AIME	
AMC	
Other	Algebra
	Number Theo
	Induction
Total	



mathd_numbertheory_227

One morning each member of Angela's
The amounts of coffee and milk varied
Angela drank a quarter of the total am
How many people are in the family?

```
Require Import Reals Lia Lra Psatz.

Theorem mathd_numbertheory_227:
  forall (x y n : nat),
    x <> 0 →
    y <> 0 →
    n <> 0 →
    (INR x / 4 + INR y / 6
      = (INR x + INR y) / INR n)%R →
    n = 5.
Proof.
  intros x y n Hx Hy Hn Heq.
  assert (Hxpos : (0 < INR x)%R) by (apply lt_0_INR; lia).
  assert (Hypos : (0 < INR y)%R) by (apply lt_0_INR; lia).
  assert (Hnpos : (0 < INR n)%R) by (apply lt_0_INR; lia).
  assert (HnR : (INR n <> 0)%R) by lra.
  assert (key : (3 * INR x * (INR n - 4) = 2 * INR y * (6 - INR n))%R).
  {
    assert (H : (12 * INR n * (INR x / 4 + INR y / 6) =
                 12 * INR n * ((INR x + INR y) / INR n))%R)
      by (f_equal; exact Heq).
    assert (H1 : (12 * INR n * (INR x / 4 + INR y / 6) =
                  3 * INR x * INR n + 2 * INR y * INR n)%R)
      by (field; lra).
    assert (H2 : (12 * INR n * ((INR x + INR y) / INR n) =
                  12 * INR x + 12 * INR y)%R)
      by (field; lra).
    lra.
  }
  destruct n as [|[|[|[|[|[|n]]]]]].
  - contradiction.
  - exfalso; rewrite S_INR, INR_0 in key; lra.
  - exfalso; rewrite !S_INR, INR_0 in key; lra.
  - exfalso; rewrite !S_INR, INR_0 in key; lra.
  - exfalso; rewrite !S_INR, INR_0 in key; lra.
  - reflexivity.
  - exfalso; rewrite !S_INR in key.
    pose proof (pos_INR n) as Hnn.
    nra.
Qed.
```

Prove Angela's family theorem in Rocq

</> ex2.v

Can you propose a proof of the theorem in @ex2.v please? Here is the informal statement of the problem: "One morning each member of Angela's family drank an

`nlinarith` isn't in scope — `Psatz` provides `nra` instead.

Edit `ex2.v`

Modified

You've used 91% of your session limit · resets in 3h · View usage

⌘ Esc to focus or unfocus Claude

ex2.v

Ask before edits

SCIENCE

AI achieves silver-medal standard solving International Mathematical Olympiad problems

Harmonic

Aristotle Achieves
Performance at the
Mathematical Olymp

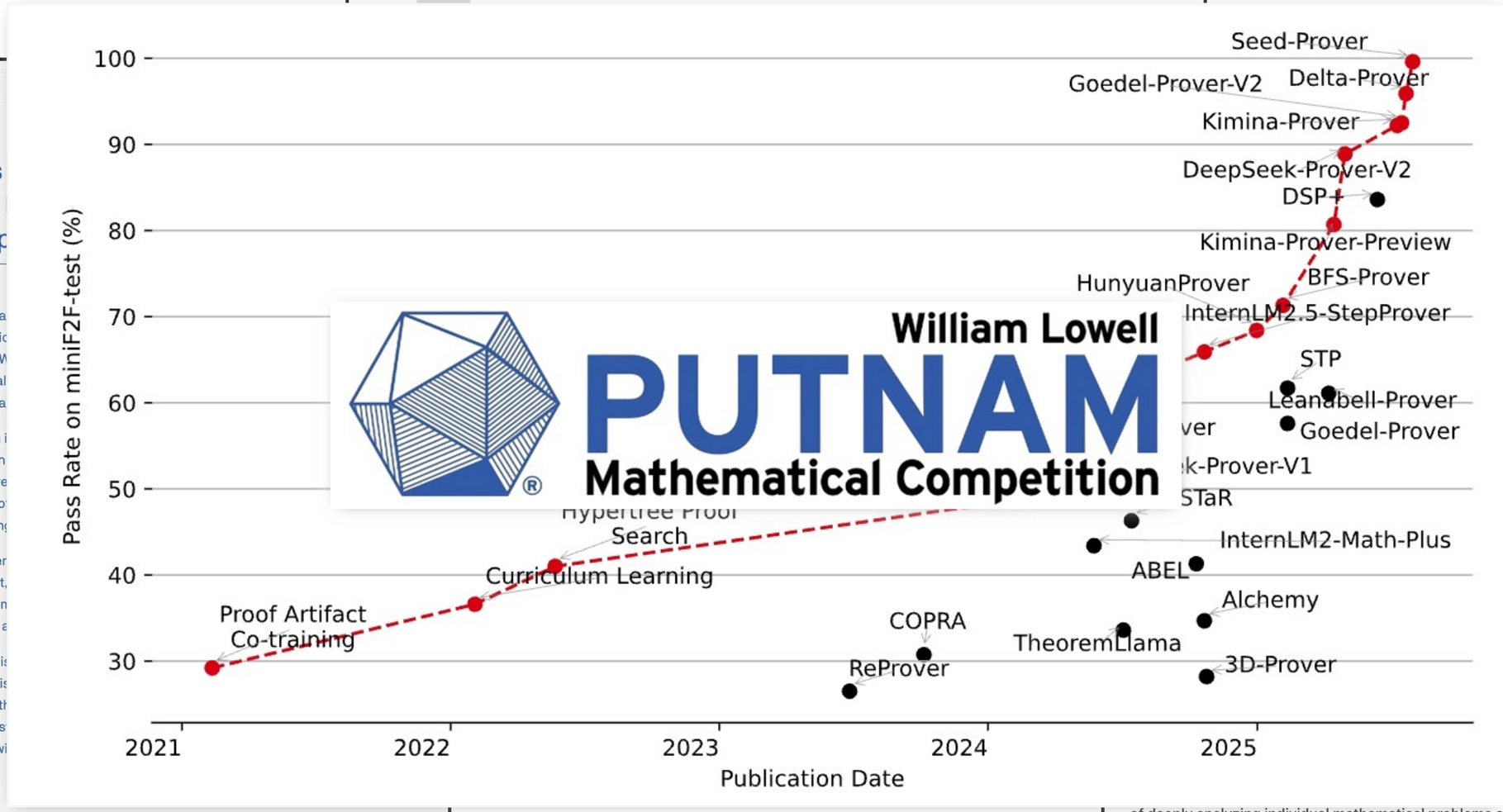
Today we announced live on X that our a
level performance at the 2025 Internatic
prestigious mathematics competition. W
human checking since they were formal
forefront of advanced mathematical rea

As the world is witnessing an explosion i
unintended side-effect of this has been
verifiers. In technical fields like software
more time confirming the correctness o
believe it is a major bottleneck in scalin

Aristotle overcomes this by design. Ever
axioms using the Lean4 proof assistant,
starting to deploy this technology with n
dependent disciplines such as physics a

As an early look into what we expect this
for iOS in Beta. When someone asks Aris
but also the corresponding Lean code th
based solving and solving multiple ques
waitlist at aristotle.harmonic.fun now, wi

Live Announcement on X
Aristotle for iOS



William Lowell
PUTNAM
Mathematical Competition

EN

ver Achieves
IMO 2025

d in IMO (The International Mathematical
atical reasoning model, Seed Prover.
ed 4 out of 6 problems and partially
quivalent to a silver medal perfor-
extended search efforts resulted in the

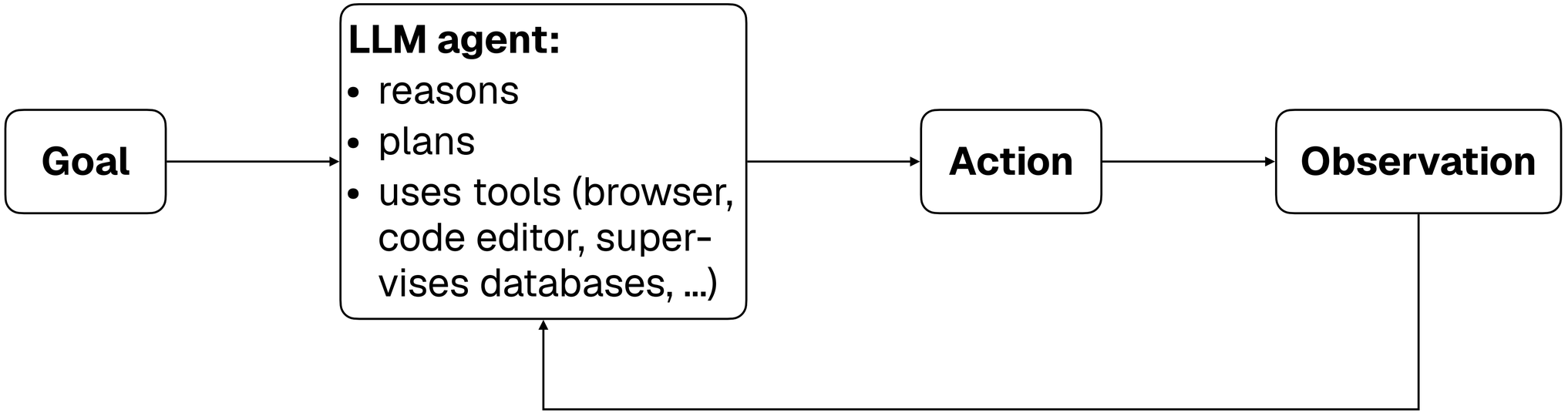
and trained using multi-stage reinforce-
hematical proofs by facilitating interac-
fied code. Through extended chain-of-
nt interaction, **Seed Prover is capable**
of deeply analyzing individual mathematical problems **continuously over several days**.

+

Reinforcement Learning

Agents

What are agents?



Classical LLM:

- one question = one answer
- no internal state
- no effects outside of the conversation

LLM agent:

- one goal leads to a sequence of actions
- maintains context overtime
- creates files, writes code, searches on the web, ...

Agents on Putnam 2025

```
/-- The sum of the first n squares formula -/
theorem sum_of_squares (n : ℕ) :
  (∑ i in Finset.range (n + 1), (i : ℚ)^2) =
    n * (n + 1) * (2 * n + 1) / 6 :=
```

Claude Code

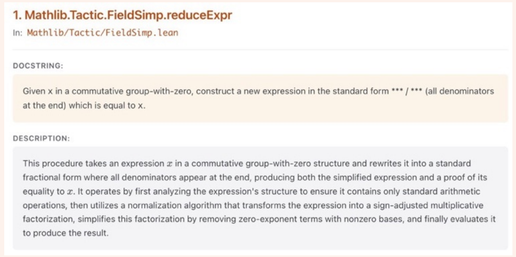
```
/-- The sum of the first n squares formula -/
theorem sum_of_squares (n : ℕ) :
  (∑ i in Finset.range (n + 1), (i : ℚ)^2) =
    n * (n + 1) * (2 * n + 1) / 6 := by
induction n with n ih
· simp
· rw [Finset.sum_range_succ, ih]
    field_simp
    ring
```

Numina-Lean-MCP

Lean-LSP-MCP

LeanDex

Simplify rational expressions with polynomials



Informal Prover

Prove the sum of squares formula for the first n natural numbers: $0^2 + 1^2 + 2^2 + \ldots + n^2 = n(n+1)(2n+1) / 6$

Detailed Proof:
1. *Base Case: n = 0 Left-Hand Side (LHS) = 0^2 = 0. Right-Hand Side (RHS) = [0(0+1)(2*0+1)] / 6 = (0 * 1 * 1) / 6 = 0. LHS = RHS. The base case holds.*
2. *Inductive Hypothesis (IH) Assume that…*

Discussion Partner

*The **ring** tactic is failing. What's missing?*

*Apply **field_simp** first. You need to clear the denominators before ring can solve the polynomial.*

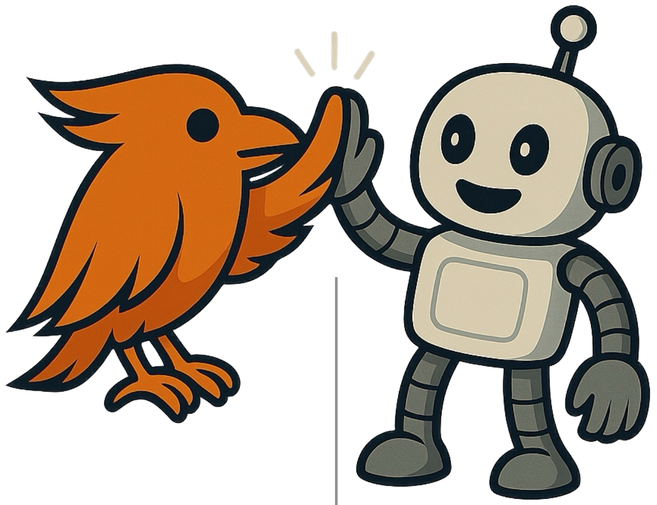
What do agents need?

coq-lsp

Flèche ⟶

petanque / pytanque

rocq-mcp



NLIR

LLM4Docq

Crrrocq

Babel

Tacq

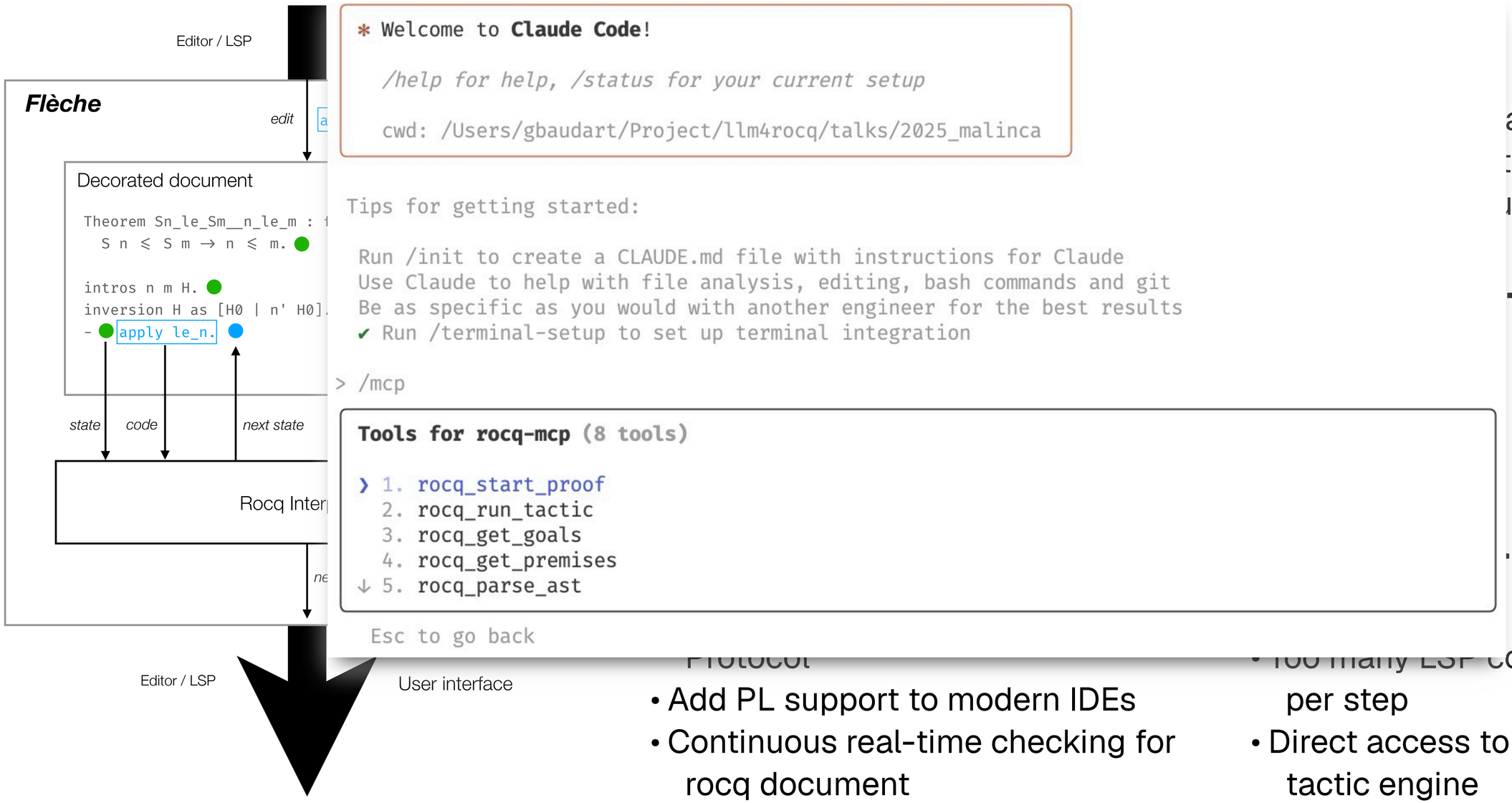
MISTRAL AI_

Claude

deepseek

ChatGPT

Interacting with Rocq



ation with
he-art LLMs
ltiple tools

-mcp

nque

are too expensive
communications

Protocol

- Add PL support to modern IDEs
- Continuous real-time checking for rocq document

- per step
- Direct access to proof state and tactic engine

Tacq:
importance of context for LLMs

Overview

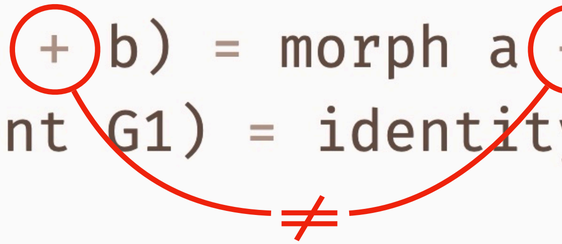
Current goal

```
R  : fieldType
h  : {poly R}
p  : {poly %/ h}
hQM  : hQ \is monic
hCp  : coprimep hQ p
p_neq0  : p ≠ 0
F  : (egcdp hQ p).1 * hQ
       + (egcdp hQ p).2 * p %= 1
```

```
((egcdp hQ p).2 * p) %% hQ %= 1 %% hQ →
(lead_coef (((egcdp hQ p).2 * p) %% hQ))^-1
  *: (((egcdp hQ p).2 * p) %% hQ)
    = 1%:P
```

Extracting notations in Rocq

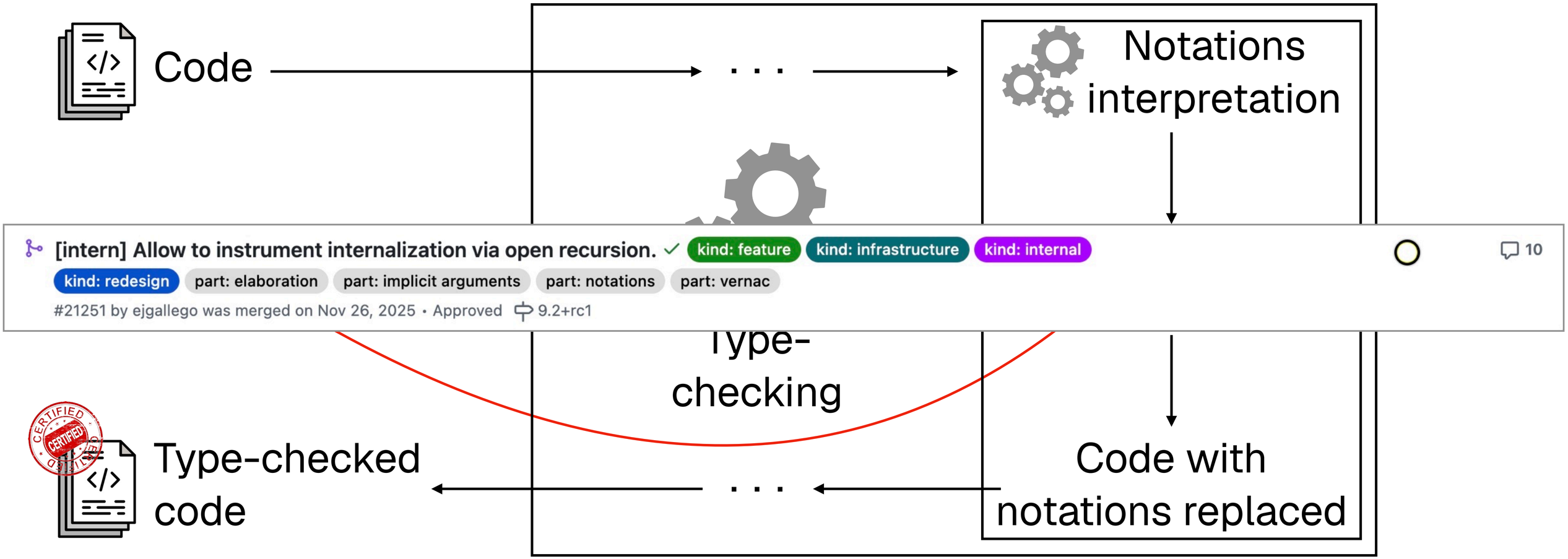
```
Record group : Type := {…
Notation "a + b" := (group_operation _ a b).

Lemma morph_id :
  forall G1 G2 : group,
  forall morph : elements G1 → elements G2,
  (forall a b, morph (a + b) = morph a + morph b) →
  morph (identity_element G1) = identity_element G2.
Proof.
```



How to differentiate them?

Extracting notations in Rocq

Modified *Rocq interpreter*



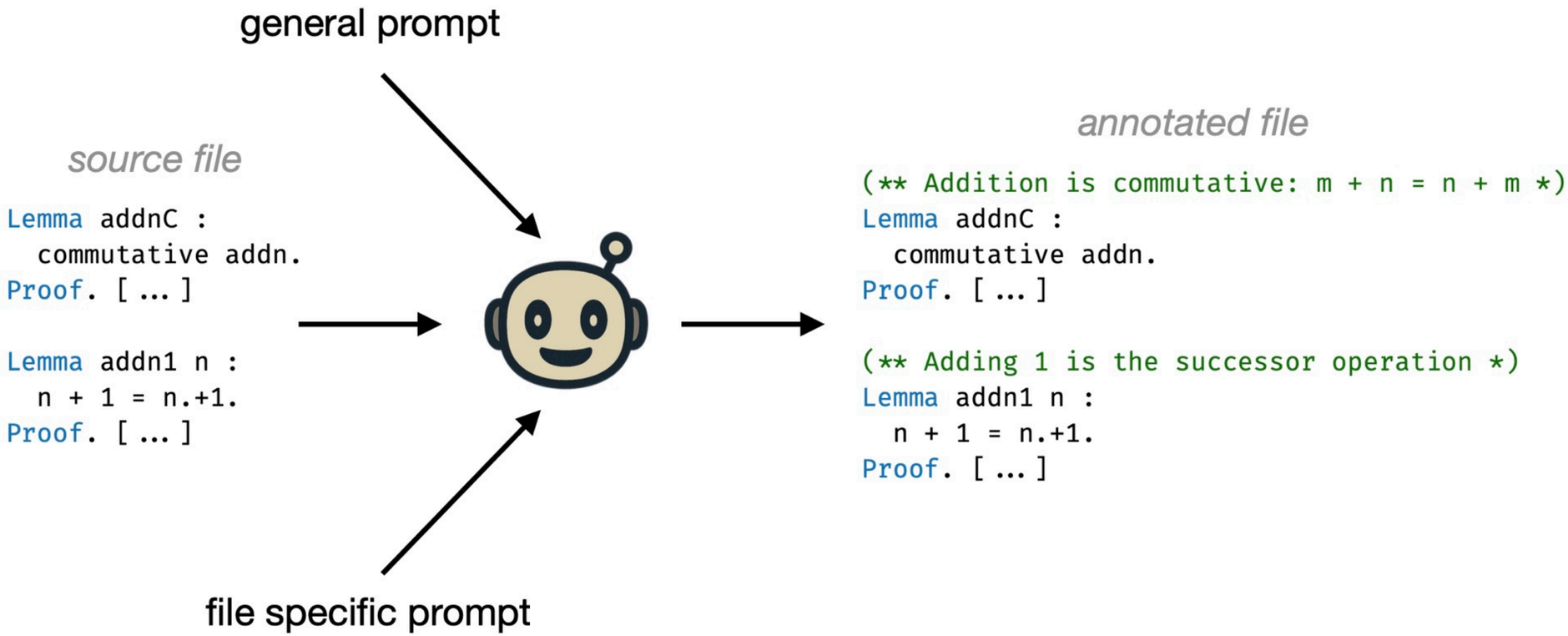
[intern] Allow to instrument internalization via open recursion. ✓ kind: feature kind: infrastructure kind: internal 10

kind: redesign part: elaboration part: implicit arguments part: notations part: vernac

#21251 by ejgallego was merged on Nov 26, 2025 • Approved 9.2+rc1

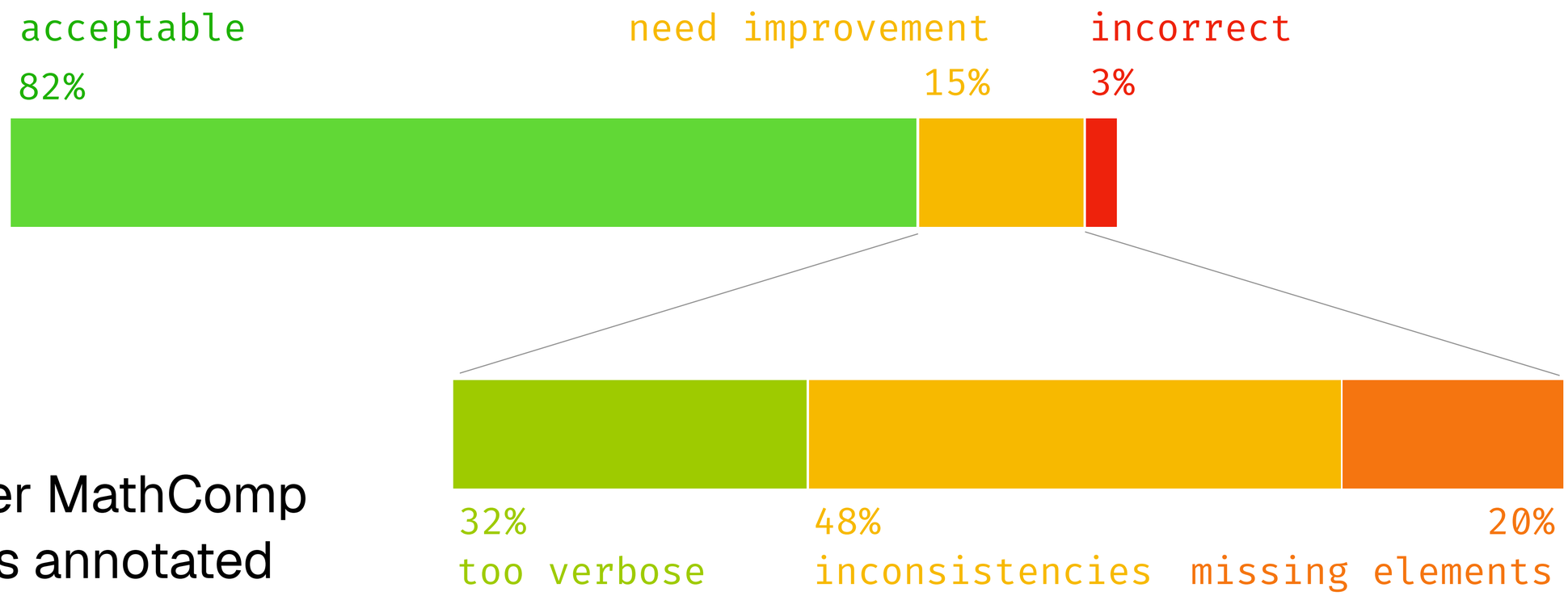
LLM4Docq: generating docstrings

MathComp: library of ~15,000 mathematical theorems in Rocq
→ use LLMs to annotate the library



LLM4Docq: generating docstrings

MathComp: library of ~15,000 mathematical theorems in Rocq
→ use LLMs to annotate the library



Results

- 2 iterations over MathComp
- 20,000 objects annotated
- 600 annotations reviewed

Experiments

- **Configurations:** *G* — *G + D + L* — *G + D & N* — *G + D & N + L*

  Example:

```
n : nat
fact_div :
  n %| factorial n
0 < n
```

```
Inductive nat : Set :=
  | O : nat
  | S : nat → nat.

Fixpoint factorial n :=
  if n is n'.+1 then n * factorial n'
  else 1.
```

```
This is an inductive type
representing the set of
natural numbers, …
```

```
This definition computes
the factorial of a natural
number, …
```

- **Pass@k:** $k \in \{1, 4, 8, 32\}$

  probability of one success among k generated samples

  Example: pass@4


```
This notation provides a
symbol for the
divisibility relation …
```

```
This notation represents
the 'strictly less than'
comparison …
```

Results

Is the generated tactic valid?

1. Bigger models benefit the most from additional context

Model	Configuration	pass@1	pass@4	pass@8	pass@32
GPT 4o	G	32.7	52.4	61.2	–
	G + D + L	35.2	55.2	64.3	–
	G + D & N	41.2	60.7	68.0	–
	G + D & N + L	**42.1**	**61.7**	**70.4**	–
Claude Sonnet 4	G	47.8	59.4	64.5	–
	G + D + L	43.5	57.7	62.8	–
	G + D & N	45.8	57.7	62.6	–
	G + D & N + L	**50.1**	**62.9**	**67.8**	–
Qwen3 32B	G	25.5	47.6	58.2	74.4
	G + D + L	27.8	50.3	60.7	77.2
	G + D & N	**29.8**	**53.2**	63.0	77.4
	G + D & N + L	29.7	53.0	**63.1**	**77.8**
Qwen3 14B	G	**26.2**	**43.4**	**51.8**	**67.1**
	G + D + L	23.8	40.9	47.5	65.1
	G + D & N	23.5	41.2	49.8	65.7
	G + D & N + L	24.1	41.8	50.4	65.7
Qwen3 4B	G	**11.1**	22.6	**30.3**	**48.5**
	G + D + L	10.9	**22.7**	30.2	47.6
	G + D & N	9.4	19.9	27.1	42.9
	G + D & N + L	11.0	22.5	29.8	46.8

Back to Putnam 2025:
do models know their needs?

Tristan Stérin

Computer Scientist

Contents

- Summary
- Setup
- Tasks

20 FEB

Tristan Stérin EDITED 19:03

With this same setup we achieved 81.1% of solved problems (198/244) on the test set of miniF2F-rocq in about 6h of active time (25h total but stalled a lot of the time because of waiting for session limits to reset).

(4) proof

t

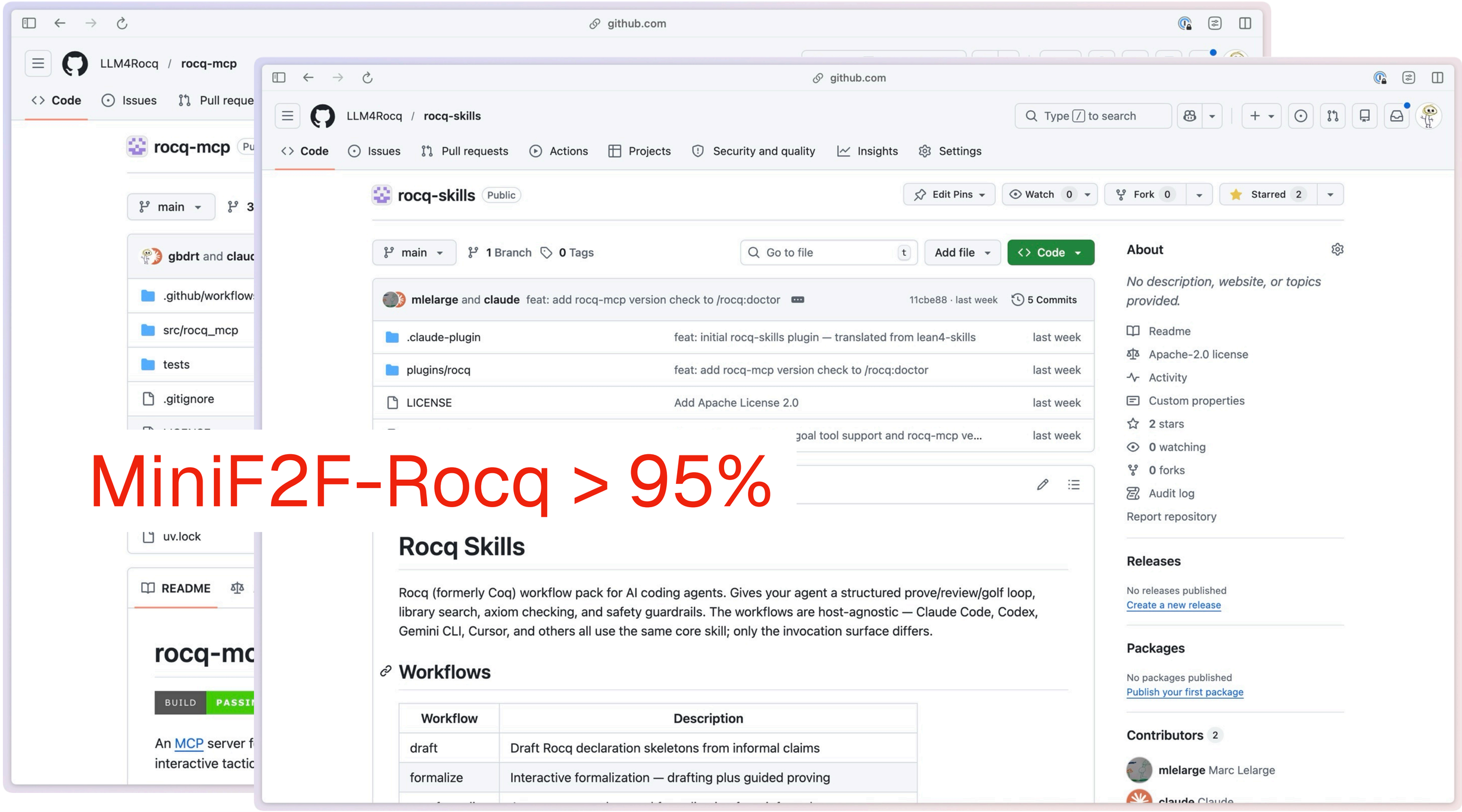
n

Ventures Rese

Opus 4.6 is great at formal proofs

February 17th 2026

What started as an excuse to try out Anthropic's new agent teams feature led me down a rabbit hole exploring how good Opus 4.6 is at formal proofs in Rocq and Lean4.


github.com
LLM4Rocq / rocq-mcp
Code
Issues
Pull reque
rocq-mcp
main
gbdrt and claud
.github/workflow
src/rocq_mcp
tests
.gitignore
uv.lock
README
rocq-mc
BUILD
PASSI
An MCP server f
interactive tactic
github.com
LLM4Rocq / rocq-skills
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security and quality
Insights
Settings
rocq-skills
Public
Edit Pins
Watch 0
Fork 0
Starred 2
main
1 Branch
0 Tags
Go to file
t
Add file
Code
mlelarge and claude feat: add rocq-mcp version check to /rocq:doctor
11cbe88 · last week
5 Commits
.claude-plugin
feat: initial rocq-skills plugin — translated from lean4-skills
last week
plugins/rocq
feat: add rocq-mcp version check to /rocq:doctor
last week
LICENSE
Add Apache License 2.0
last week
goal tool support and rocq-mcp ve...
last week
MiniF2F-Rocq > 95%
Rocq Skills
Rocq (formerly Coq) workflow pack for AI coding agents. Gives your agent a structured prove/review/golf loop, library search, axiom checking, and safety guardrails. The workflows are host-agnostic — Claude Code, Codex, Gemini CLI, Cursor, and others all use the same core skill; only the invocation surface differs.
Workflows
Workflow
Description
draft
Draft Rocq declaration skeletons from informal claims
formalize
Interactive formalization — drafting plus guided proving
About
No description, website, or topics provided.
Readme
Apache-2.0 license
Activity
Custom properties
2 stars
0 watching
0 forks
Audit log
Report repository
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
mlelarge Marc Lelarge
claude Claude

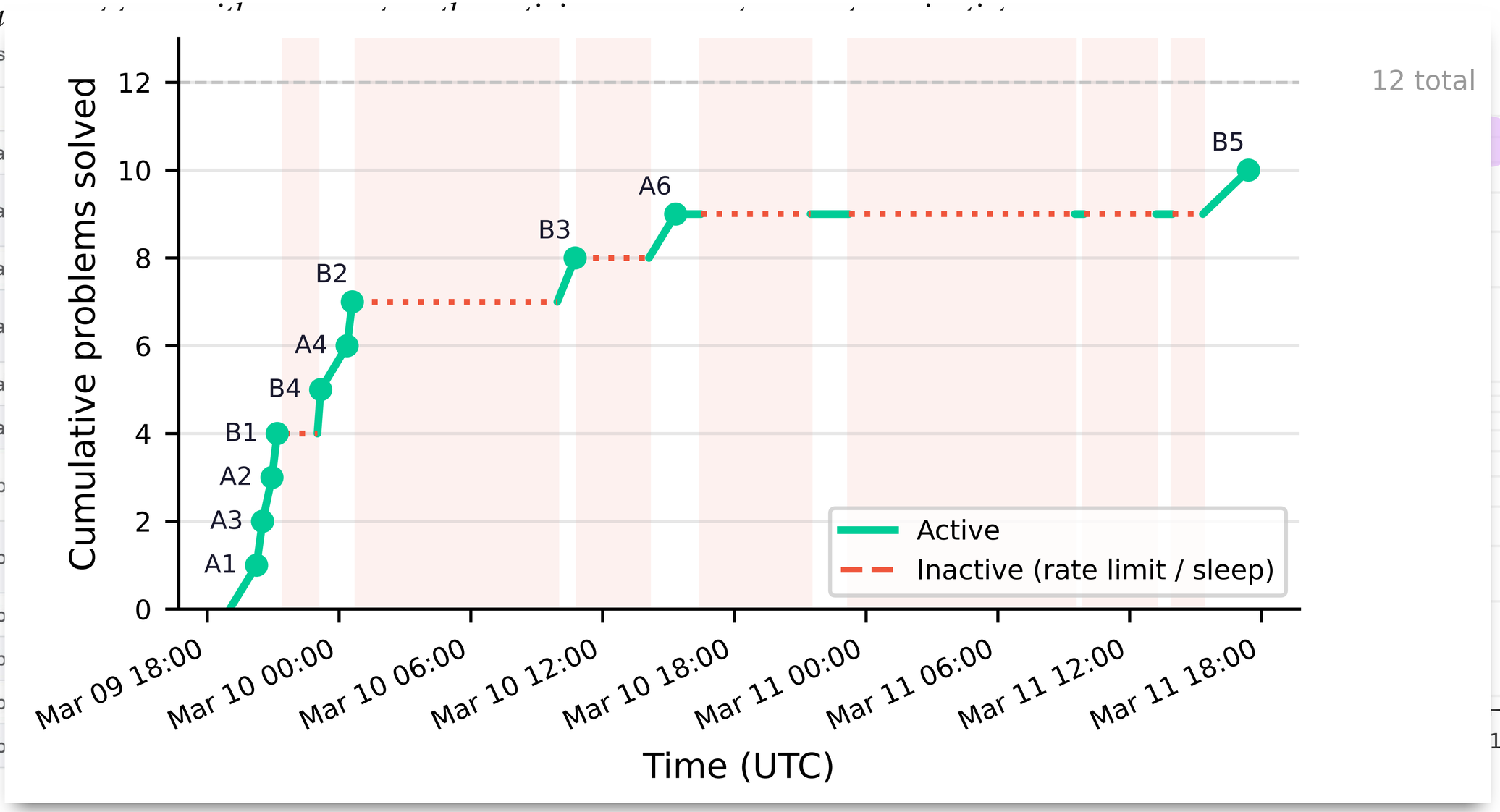
Results

Launch a

Formal statements

Problem
putnam_2025_a
putnam_2025_a
putnam_2025_a
putnam_2025_a
putnam_2025_a
putnam_2025_a
putnam_2025_b
putnam_2025_b
putnam_2025_b
putnam_2025_b
putnam_2025_b
putnam_2025_b

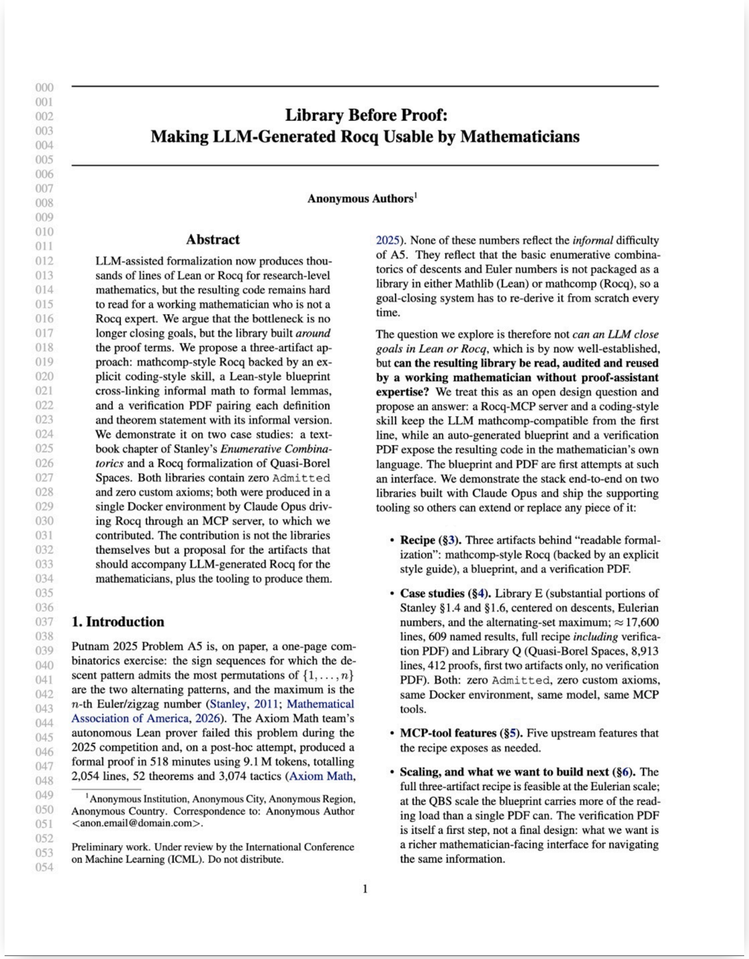
10/12 verified



https://github.com/LLM4Rocq/Putnam2025-Rocq

Targeting the remaining problems

- **B6**: solved when given the Lean proof
- **A5**: lead to the creation of a whole library:
  mathcomp-eulerian
  - Claude Opus 4.6 / 4.7 + rocq-mcp
  - exact correspondence with sections 1.4 and 1.6 of Stanley's *Enumerative Combinatorics*
  - 33 files ≈ 17,600 lines
  - following the style of *mathcomp*, the biggest mathematical library in Rocq



Library Before Proof:
Making LLM-Generated Rocq Usable by Mathematicians

Anonymous Authors[1]

Abstract

LLM-assisted formalization now produces thousands of lines of Lean or Rocq for research-level mathematics, but the resulting code remains hard to read for a working mathematician who is not a Rocq expert. We argue that the bottleneck is no longer closing goals, but the library built *around* the proof terms. We propose a three-artifact approach: mathcomp-style Rocq backed by an explicit coding-style skill, a Lean-style blueprint cross-linking informal math to formal lemmas, and a verification PDF pairing each definition and theorem statement with its informal version. We demonstrate it on two case studies: a textbook chapter of Stanley's *Enumerative Combinatorics* and a Rocq formalization of Quasi-Borel Spaces. Both libraries contain zero `Admitted` and zero custom axioms; both were produced in a single Docker environment by Claude Opus driving Rocq through an MCP server, to which we contributed. The contribution is not the libraries themselves but a proposal for the artifacts that should accompany LLM-generated Rocq for the mathematicians, plus the tooling to produce them.

1. Introduction

Putnam 2025 Problem A5 is, on paper, a one-page combinatorics exercise: the sign sequences for which the descent pattern admits the most permutations of $\{1, \ldots, n\}$ are the two alternating patterns, and the maximum is the n-th Euler/zigzag number (Stanley, 2011; Mathematical Association of America, 2026). The Axiom Math team's autonomous Lean prover failed this problem during the 2025 competition and, on a post-hoc attempt, produced a formal proof in 518 minutes using 9.1 M tokens, totalling 2,054 lines, 52 theorems and 3,074 tactics (Axiom Math, 2025). None of these numbers reflect the *informal* difficulty of A5. They reflect that the basic enumerative combinatorics of descents and Euler numbers is not packaged as a library in either Mathlib (Lean) or mathcomp (Rocq), so a goal-closing system has to re-derive it from scratch every time.

The question we explore is therefore not *can an LLM close goals in Lean or Rocq*, which is by now well-established, but **can the resulting library be read, audited and reused by a working mathematician without proof-assistant expertise?** We treat this as an open design question and propose an answer: a Rocq-MCP server and a coding-style skill keep the LLM mathcomp-compatible from the first line, while an auto-generated blueprint and a verification PDF expose the resulting code in the mathematician's own language. The blueprint and PDF are first attempts at such an interface. We demonstrate the stack end-to-end on two libraries built with Claude Opus and ship the supporting tooling so others can extend or replace any piece of it:

- **Recipe (§3).** Three artifacts behind "readable formalization": mathcomp-style Rocq (backed by an explicit style guide), a blueprint, and a verification PDF.
- **Case studies (§4).** Library E (substantial portions of Stanley §1.4 and §1.6, centered on descents, Eulerian numbers, and the alternating-set maximum; ≈ 17,600 lines, 609 named results, full recipe *including* verification PDF) and Library Q (Quasi-Borel Spaces, 8,913 lines, 412 proofs, first two artifacts only, no verification PDF). Both: zero `Admitted`, zero custom axioms, same Docker environment, same model, same MCP tools.
- **MCP-tool features (§5).** Five upstream features that the recipe exposes as needed.
- **Scaling, and what we want to build next (§6).** The full three-artifact recipe is feasible at the Eulerian scale; at the QBS scale the blueprint carries more of the reading load than a single PDF can. The verification PDF is itself a first step, not a final design: what we want is a richer mathematician-facing interface for navigating the same information.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

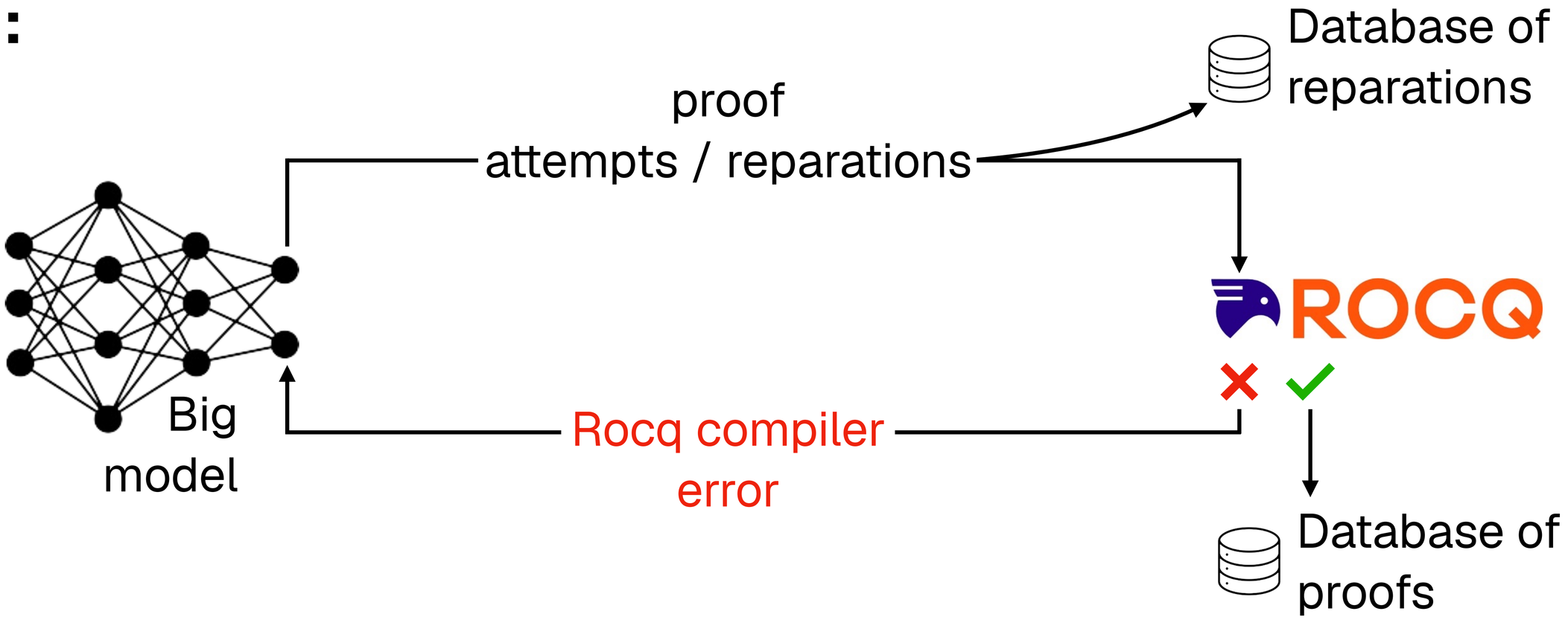
Proof agent:
impact of good examples

Distillation by error recovery

Distillation: transferring knowledge from a bigger model to a smaller one

Goal: *limit the number of calls to big models*

Run 1:

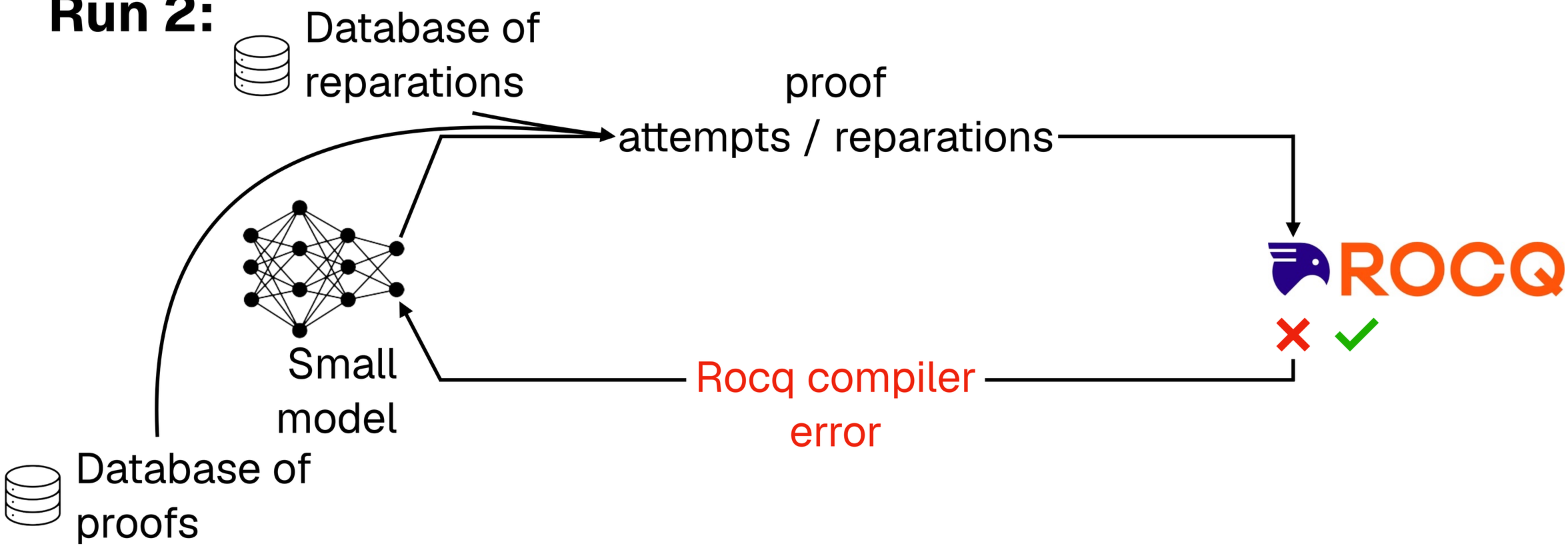


Distillation by error recovery

Distillation: transferring knowledge from a bigger model to a smaller one

Goal: *limit the number of calls to big models*

Run 2:



Introduction to embedding models

Question: *can models represent meaning?*

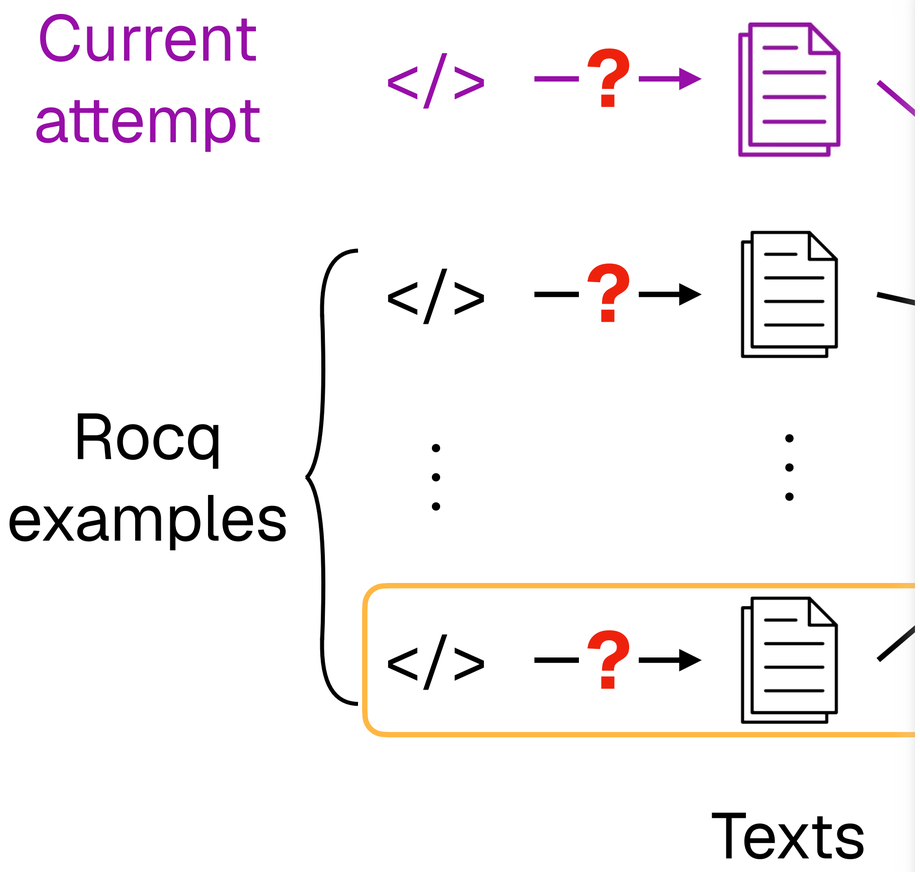
The meaning is **implicit** in LLMs

- The mouse ate the … cheese ✓ / cat ✗ ⟶ choose the most probable
- Learning about the probability = learning about the meaning
- Inside LLMs: words = vectors ⟶ turn **language into geometry**

king - man + woman ≈ queen

Embedding models make the meaning **explicit**
= they return vectors representing the meaning of a sentence

Retrieving relevant examples



Successful proofs:

- natural language description of the problem
- Rocq statement of the problem

Reparation examples:

- Rocq compiler error
- failing tactic
- proof up until the failing tactic

Results

Models:

- Big
- Small
- Embedd

Run 1 on

5 successful proofs examples +

5 reparation examples at each error

Thank you!

Contact: jules.viennot@gmail.com

GitHub: https://github.com/LLM4Rocq