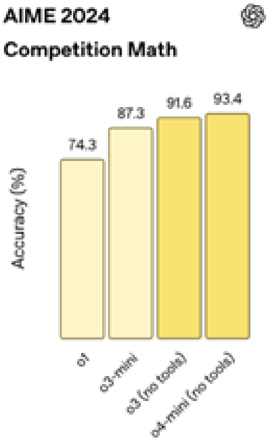
Intro to AI for Mathematics

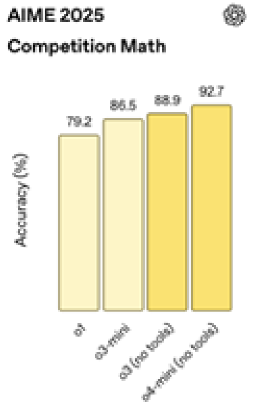
Jialin (Mike) Lu

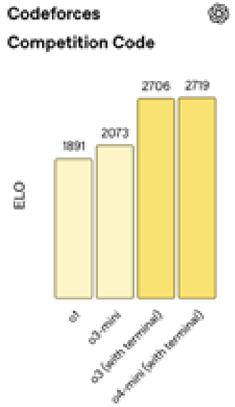
Frontier LLMs Competing in Math and Coding

Gemini 3

AIME 2025	Mathematics	No tools	95.0%
		With code execution	100%
MathArena Apex	Challenging Math Contest problems		23.4%
LiveCodeBench Pro	Competitive coding problems from Codeforces, ICPC, and IOI	Elo Rating, higher is better	2,439
Terminal-Bench 2.0	Agentic terminal coding	Terminus-2 agent	54.2%
SWE-Bench Verified	Agentic coding	Single attempt	76.2%







Why Math and Coding?

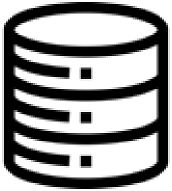
- Proxies for **complex reasoning** and **planning**
  - Important for human intelligence; challenging for LLMs
- Relatively easy to evaluate
  - Math: just check answers
  - Coding: run some unit tests
  - Writing a crime fiction? Composing a symphony?

How LLMs are Trained to Solve Math Problems?

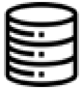
- Base LLM + **two ingredients** + engineering
- **Supervised fine-tuning (SFT):** “**Good data** is all you need”
- **Reinforcement learning (RL):** “**Verifiability** is all you need”

Supervised Finetuning on Mathematical Data

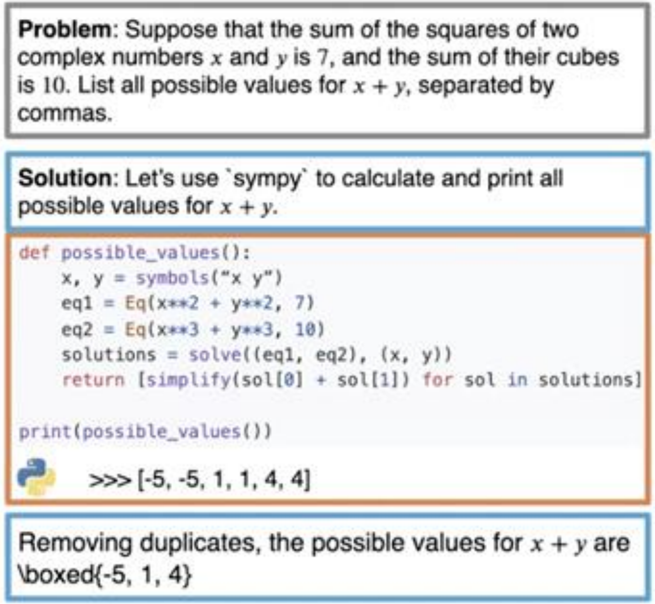
Math related web documents



Problems w/ step by step solutions

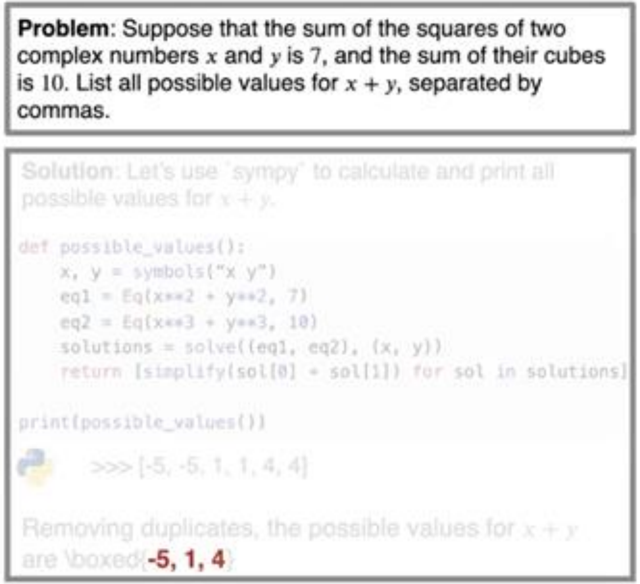


Problems w/ t
integrated solutions



Supervised Finetuning on Mathematical Data

Learn from final answer?


Problem: Suppose that the sum of the squares of two complex numbers x and y is 7, and the sum of their cubes is 10. List all possible values for $x + y$, separated by commas.

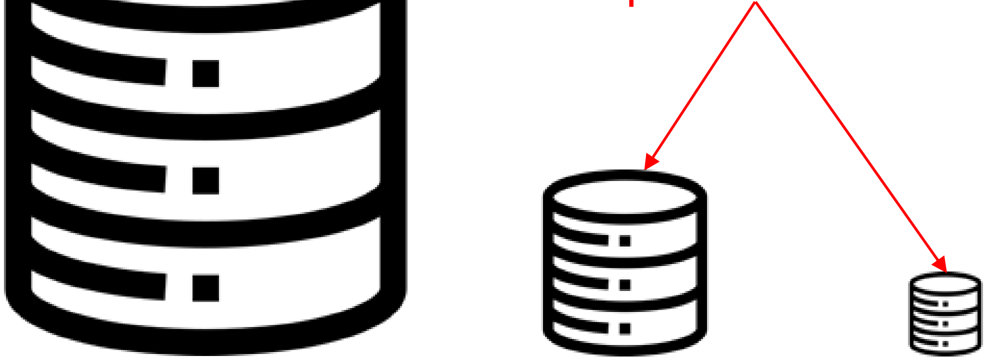
Solution: Let's use `sympy` to calculate and print all possible values for $x + y$.

```
def possible_values():
    x, y = symbols("x y")
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1, eq2), (x, y))
    return [simplify(sol[0] + sol[1]) for sol in solutions]

print(possible_values())
```

>>> [-5, -5, 1, 1, 4, 4]

Removing duplicates, the possible values for $x + y$ are \boxed{**-5, 1, 4**}

Expensive to collect

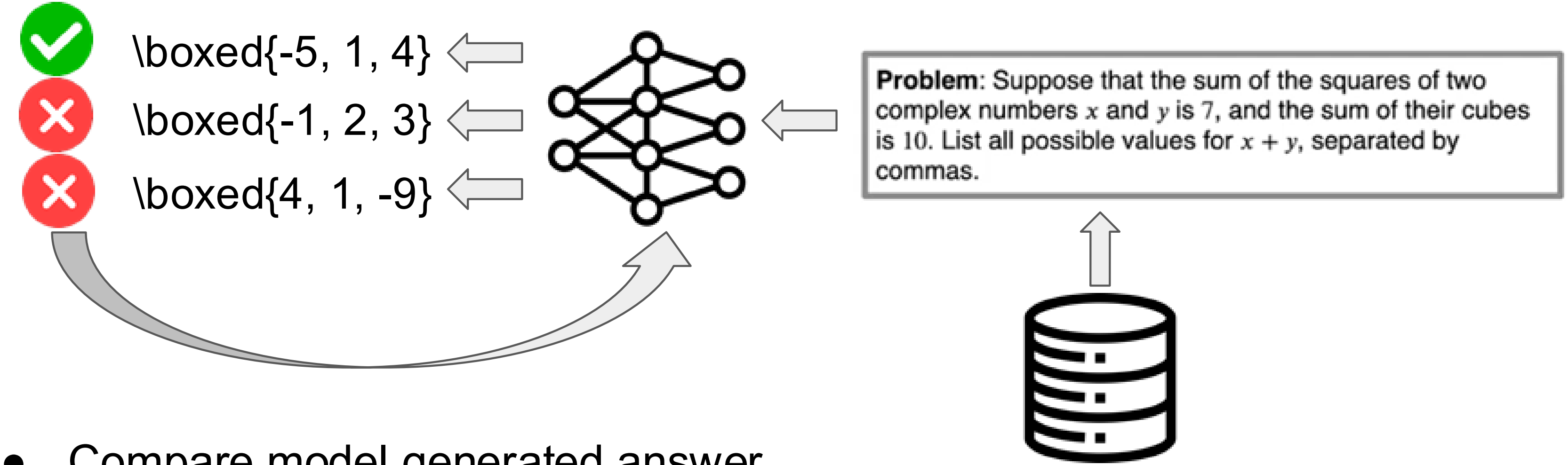


Math related web documents

Problems w/ step by step solutions

Problems w/ tool integrated solutions

Reinforcement Learning on Verifiable Problems



Math problems w/o step by step solutions

- Compare model generated answer with ground truth to assign reward
- Use GRPO to optimize LLM to achieve high rewards
- **But not applicable to proofs?**

Proofs are Hard to Verify

We will show that if

$$a, b, c > 0 \quad \text{and} \quad abc = 1, \tag{66}$$

then

$$\frac{1}{c^3(a+b)} + \frac{1}{a^3(b+c)} + \frac{1}{b^3(c+a)} \geq \frac{3}{2}. \tag{67}$$

There are several ways to prove this inequality. One very neat method is to use the well-known inequality

$$a + b \geq 2\sqrt{ab}, \tag{68}$$

together with the fact that $abc = 1$. (Any correct solution is acceptable.) We now describe one complete solution.

Step 1. Apply the AM–GM Inequality to the Denominators

Let us start by looking at the first term:

$$\frac{1}{c^3(a+b)}. \tag{69}$$

Since $a + b \geq 2\sqrt{ab}$, we have

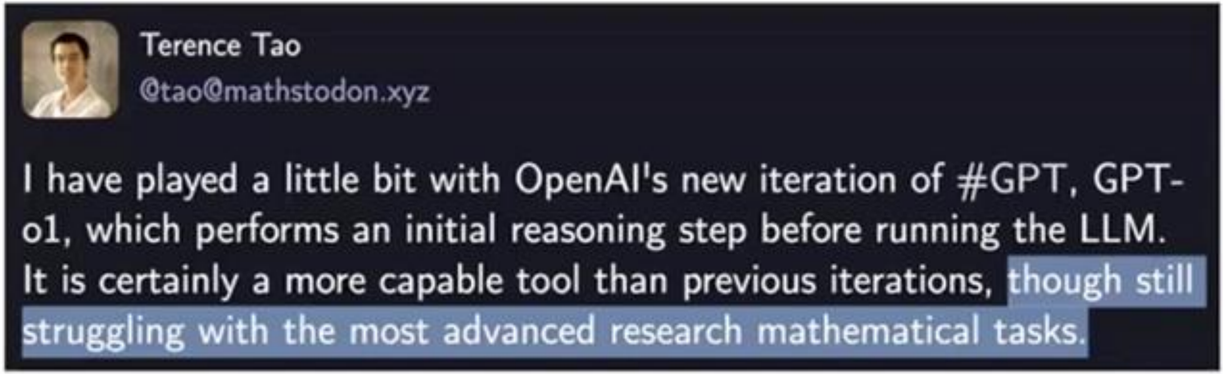
$$\frac{1}{a+b} \leq \frac{1}{2\sqrt{ab}}. \tag{70}$$

Thus,

$$\frac{1}{c^3(a+b)} \geq \frac{1}{c^3} \cdot \frac{1}{2\sqrt{ab}} = \frac{1}{2\,c^3\sqrt{ab}}. \tag{71}$$

Beyond High School Math

- Advanced math mostly involve proofs
- Existing LLM success are mostly on high-school level concepts
- LLMs struggle with advanced math (research-level)



Problems

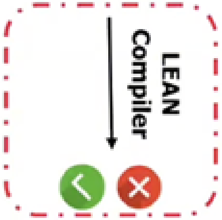
- Proofs are important for research-level math
- Data for advanced math are scarce
- Proofs can’t be reliably verified, so RL fails
- What can we do?

Key Ingredient: Formal Reasoning

- **Formal reasoning**: ground mathematical reasoning in formal systems
- **Formal system**: can verify proofs and provide ground truth feedback
- Integrating LLM with formal reasoning enables learning from feedback
- Enables verification of proofs and reasoning

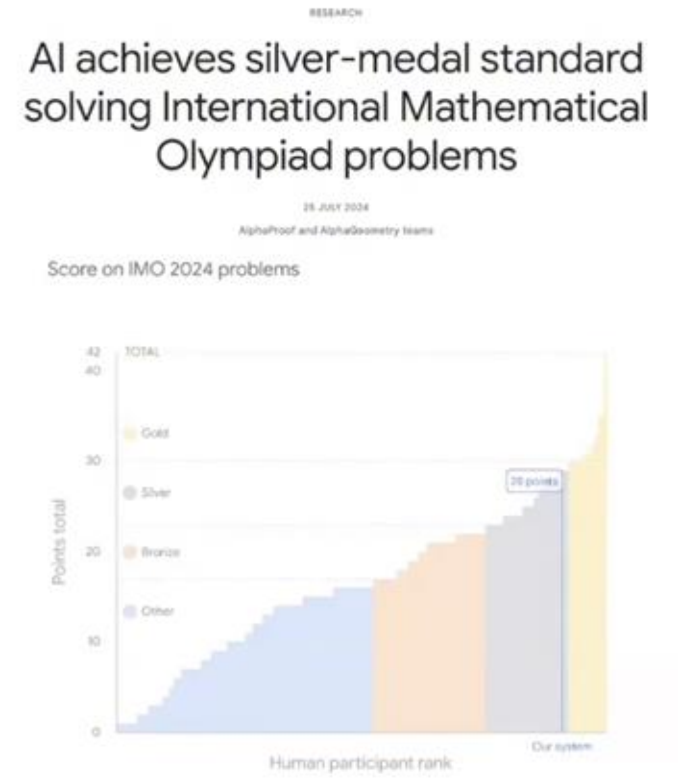
Proof Assistants

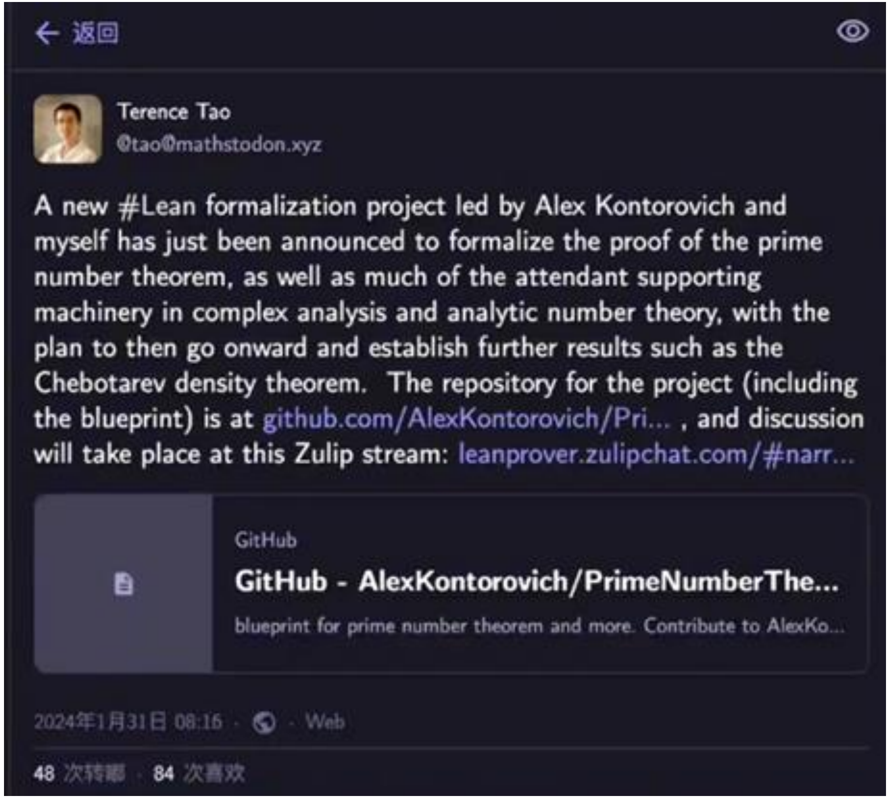
```
import Mathlib

def count_divisible_by_10 : ℕ :=
  (finset.range 6).card * (finset.range 6).card * (finset.range 6).card

def is_divisible_by_10 (x y z : ℕ) : Prop :=
  x * y * z % 10 = 0

theorem count_cases_divisible_by_10 :
  let valid_numbers := (finset.range 1 7) in
  valid_numbers.filter (λ x, valid_numbers.filter (λ y, valid_numbers.filter (λ z, is_divisible_by_10
  x y z)).card).sum = 135 :=
begin
  -- Count the valid cases
  have h : (finset.range 1 7).prod (λ x, (finset.range 1 7).prod (λ y, (finset.range 1 7).filter (λ z,
  is_divisible_by_10 x y z).card)) = 135,
  { sorry }, -- Proof goes here
  exact h,
end
```



Machine Checkable!

Impacts in Math





LLM + Formal Mathematics

Theorem

<u>There exists an infinite number of primes</u>

⇩ Autoformalization

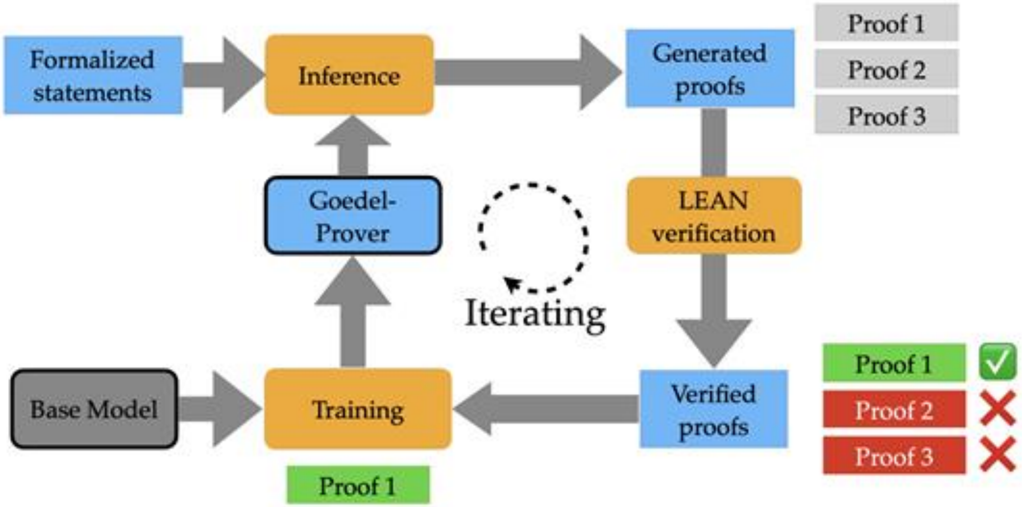
```
theorem exists_infinite_primes (n : ℕ) : ∃ p, n ≤ p ∧ Prime p
```

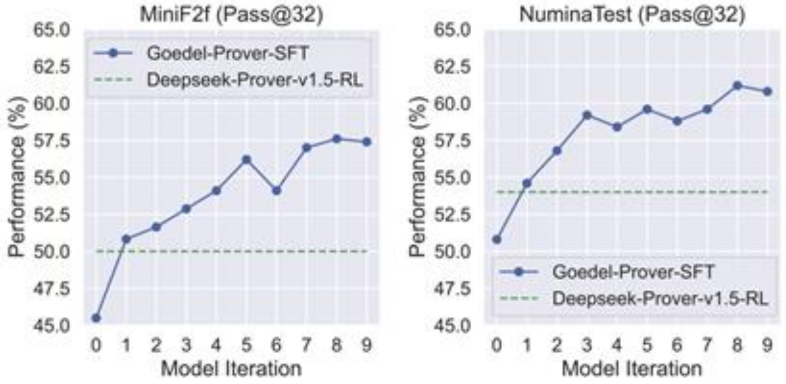
⇩ Theorem Proving

```
let p := minFac (n ! + 1)
have f1 : n ! + 1 ≠ 1 := ne_of_gt <| succ_lt_succ <| factorial_pos _
have pp : Prime p := minFac_prime f1
have np : n ≤ p :=
  le_of_not_ge fun h =>
    have h₁ : p ∣ n ! := dvd_factorial (minFac_pos _) h
    have h₂ : p ∣ 1 := (Nat.dvd_add_iff_right h₁).2 (minFac_dvd _)
    pp.not_dvd_one h₂
(p, np, pp)
```

Goedel-Prover

A New Frontier in Open-source Automated Theorem Proving





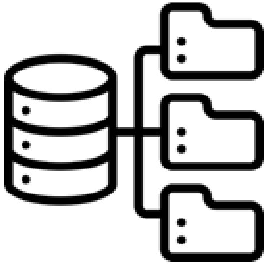
- Follow up works uses test time approaches, RL…

Limitations

- Theorem proving relies on using predefined statements/concepts
- Previous work assumes LLM memorizes the library

```
/-- **Property 2**: The heat kernel integrates to one over all of ℝ.

  This shows that the heat kernel is properly normalized and can be interpreted
  as a probability density function (specifically, a Gaussian distribution). -/
lemma integral_heatKernel_one_gaussian (hα : 0 < α) (ht : 0 < t) :
  ∫ x : ℝ, heatKernel α x t = 1 := by
  let b := 1 / (4 * α * t)
  have b_pos : 0 < b := by positivity
  calc ∫ x : ℝ, heatKernel α x t
    = ∫ x : ℝ, (1 / Real.sqrt (Real.pi / b)) * Real.exp (-b * x^2) := by
      congr 1; funext x; simp [heatKernel, b, div_eq_mul_inv]; ring_nf
    _ = (1 / Real.sqrt (Real.pi / b)) * ∫ x : ℝ, Real.exp (-b * x^2) :=
      integral_const_mul _ _
    _ = (1 / Real.sqrt (Real.pi / b)) * Real.sqrt (Real.pi / b) := by
      rw [integral_gaussian b]
    _ = 1 := by field_simp [ne_of_gt b_pos]
```

>300K statements

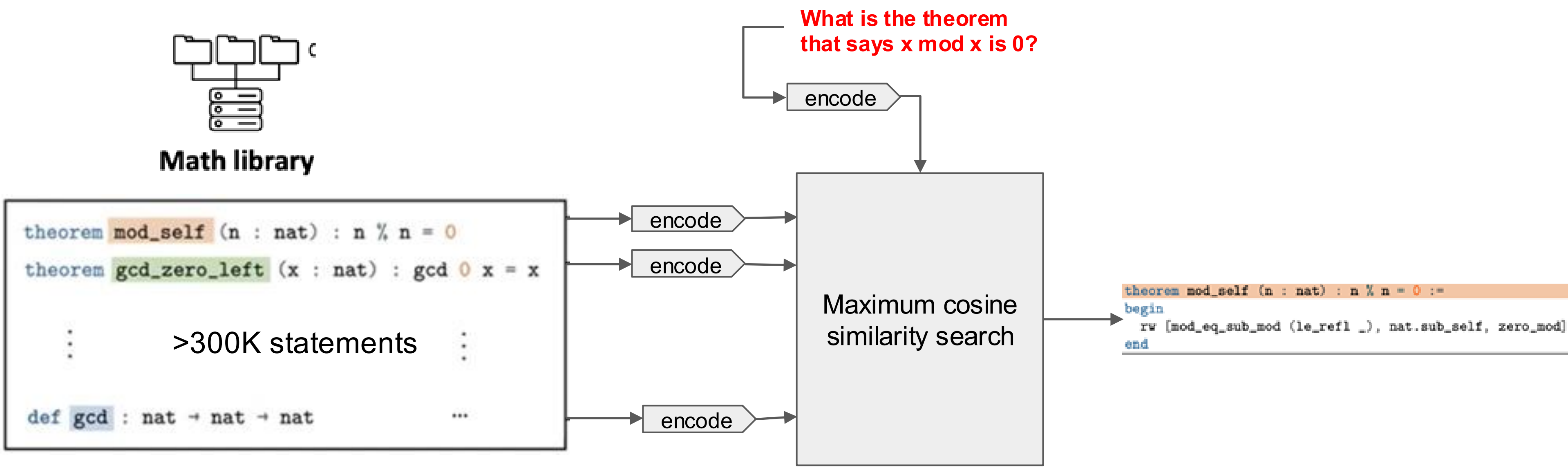


Mathlib

```
theorem div_eq_mul_inv (a b : G) : a / b = a * b⁻¹ :=
  DivInvMonoid.div_eq_mul_inv _ _
```

```
noncomputable def sqrt (x : ℝ) : ℝ :=
  NNReal.sqrt (Real.toNNReal x)
```

LEAN FINDER: SEMANTIC SEARCH FOR MATHLIB THAT UNDERSTANDS USER INTENTS



- Boost LLM-prover performance, adopted by mathematicians, integrated in many agent workflows

https://www.leanfinder.org/

Takeaways

- LLMs struggle in advanced mathematics due to data scarcity and difficulty to verify
- Formal systems verify absolute correctness of proofs, provide learning signals
- Training techniques like Expert Iteration & RL takes advantage of verifiability
- **Ultimate Goal**: verified reasoning that bridge natural language with formal verification

References

Goedel Prover: https://arxiv.org/abs/2502.07640

Lean Finder: https://arxiv.org/pdf/2510.15940

Some slides are adapted from Kaiyu Yang's talk