

# Intro to *AI* for Mathematics

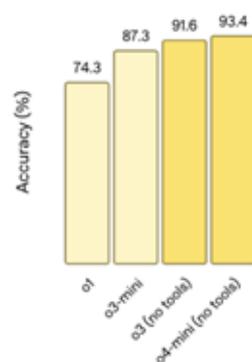
Jialin (Mike) Lu

# Frontier LLMs Competing in Math and Coding

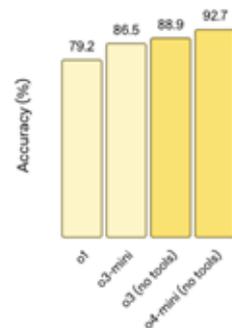


AIME 2025	Mathematics	No tools	95.0%
		With code execution	100%
MathArena Apex	Challenging Math Contest problems		23.4%
LiveCodeBench Pro	Competitive coding problems from Codeforces, ICPC, and IOI	Elo Rating, higher is better	2,439
Terminal-Bench 2.0	Agentic terminal coding	Terminus-2 agent	54.2%
SWE-Bench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

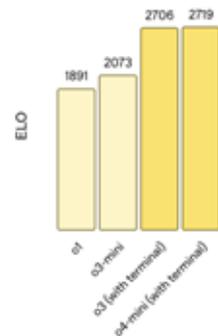
AIME 2024  
Competition Math



AIME 2025  
Competition Math



Codeforces  
Competition Code



# 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/ integrated solutions

**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}`

# Supervised Finetuning on Mathematical Data

Learn from final answer?



Math related web documents

Expensive to collect



Problems w/ step by step solutions



Problems w/ terse integrated solutions

**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.

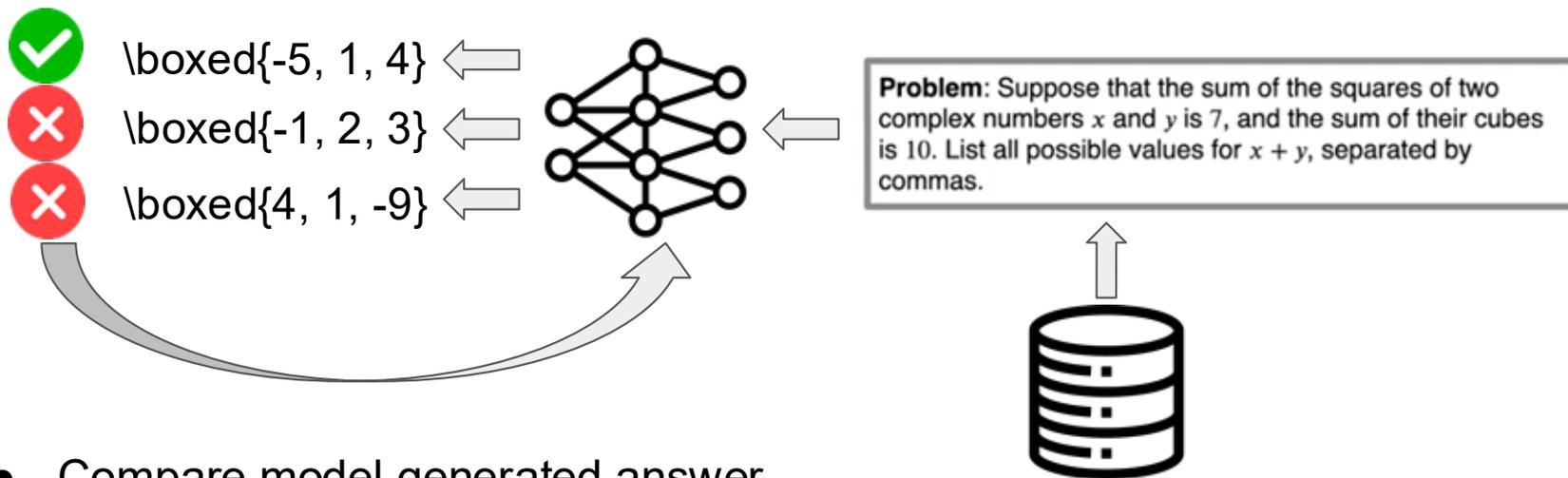
**Solution:** Let's use 'sympy' to calculate and print all possible values for  $x + y$ .

```
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 `{-5, 1, 4}`.

# Reinforcement Learning on Verifiable Problems



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

Math problems w/o step by step solutions

# Proofs are Hard to Verify

We will show that if

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

then

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

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

$$a + b \geq 2\sqrt{ab}, \quad (68)$$

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:

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

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

$$\frac{1}{a+b} \leq \frac{1}{2\sqrt{ab}}. \quad (70)$$

Thus,

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

# 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)



Terence Tao

@tao@mathstodon.xyz

I have played a little bit with OpenAI's new iteration of #GPT, GPT-o1, which performs an initial reasoning step before running the LLM. It is certainly a more capable tool than previous iterations, though still struggling with the most advanced research mathematical tasks.

# 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?

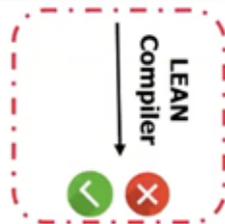
# Key Ingredient: Formal Reasoning

- **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



```
1 import Mathlib
2
3 def count_divisible_by_10 : N :=
4   (finset.range 6).card * (finset.range 6).card * (finset.range 6).card
5
6 def is_divisible_by_10 (x y z : N) : Prop :=
7   x * y * z % 10 = 0
8
9 theorem count_cases_divisible_by_10 :
10  let valid_numbers := (finset.range 1 7) in
11  valid_numbers.filter (λ x, valid_numbers.filter (λ y, valid_numbers.filter (λ z, is_divisible_by_10
12  x y z))).card).sum = 135 :=
13  begin
14    -- Count the valid cases
15    have h : (finset.range 1 7).prod (λ x, (finset.range 1 7).prod (λ y, (finset.range 1 7).filter (λ z,
16    is_divisible_by_10 x y z).card)) = 135,
17    { sorry }, -- Proof goes here
18    exact h,
19  end
```



Machine Checkable!

# Impacts in Math

## AI achieves silver-medal standard solving International Mathematical Olympiad problems

RESEARCH

25 JULY 2024

AlphaProof and AlphaGeometry teams

Score on IMO 2024 problems



← 返回



Terence Tao

@tao@mathstodon.xyz

A new #Lean formalization project led by Alex Kontorovich and myself has just been announced to formalize the proof of the prime number theorem, as well as much of the attendant supporting machinery in complex analysis and analytic number theory, with the plan to then go onward and establish further results such as the Chebotarev density theorem. The repository for the project (including the blueprint) is at [github.com/AlexKontorovich/Pri...](https://github.com/AlexKontorovich/PrimeNumberTheorem), and discussion will take place at this Zulip stream: [leanprover.zulipchat.com/#narr...](https://leanprover.zulipchat.com/#narr...)



GitHub

**GitHub - AlexKontorovich/PrimeNumberThe...**

blueprint for prime number theorem and more. Contribute to AlexKo...

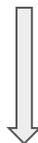
2024年1月31日 08:16 · 🌐 · Web

48 次转聊 · 84 次喜欢

# LLM + Formal Mathematics

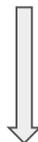
## Theorem

There exists an infinite number of primes



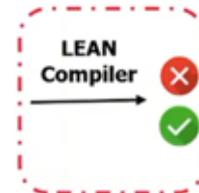
Autoformalization

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



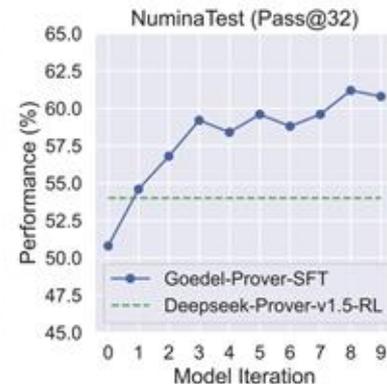
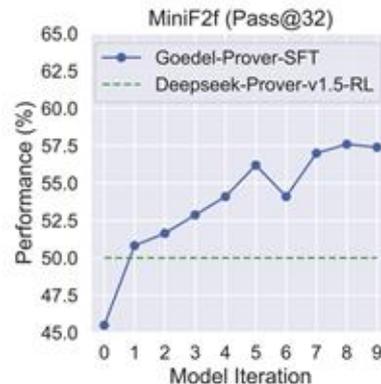
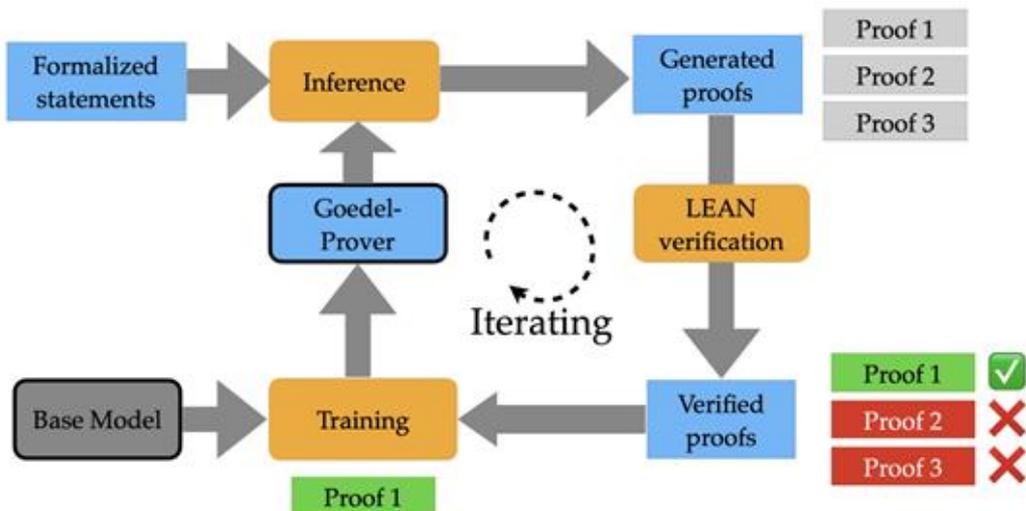
Theorem Proving

```
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

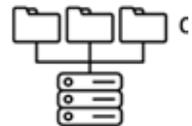


**Mathlib**

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

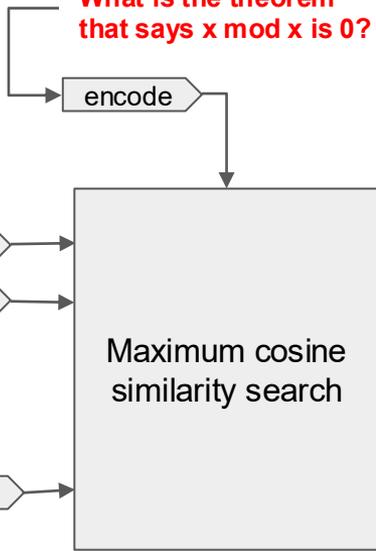
```
noncomputable def sqrt (x : R) : R :=  
  NNReal.sqrt (Real.toNNReal x)
```

# LEAN FINDER: SEMANTIC SEARCH FOR MATHLIB THAT UNDERSTANDS USER INTENTS



Math library

What is the theorem that says  $x \bmod x$  is 0?



```
theorem mod_self (n : nat) : n % n = 0
theorem gcd_zero_left (x : nat) : gcd 0 x = x
...
>300K statements
...
def gcd : nat → nat → nat ...
```

```
theorem mod_self (n : nat) : n % n = 0 :=
begin
  rw [mod_eq_sub_mod (le_refl _), nat.sub_self, zero_mod]
end
```

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

# 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