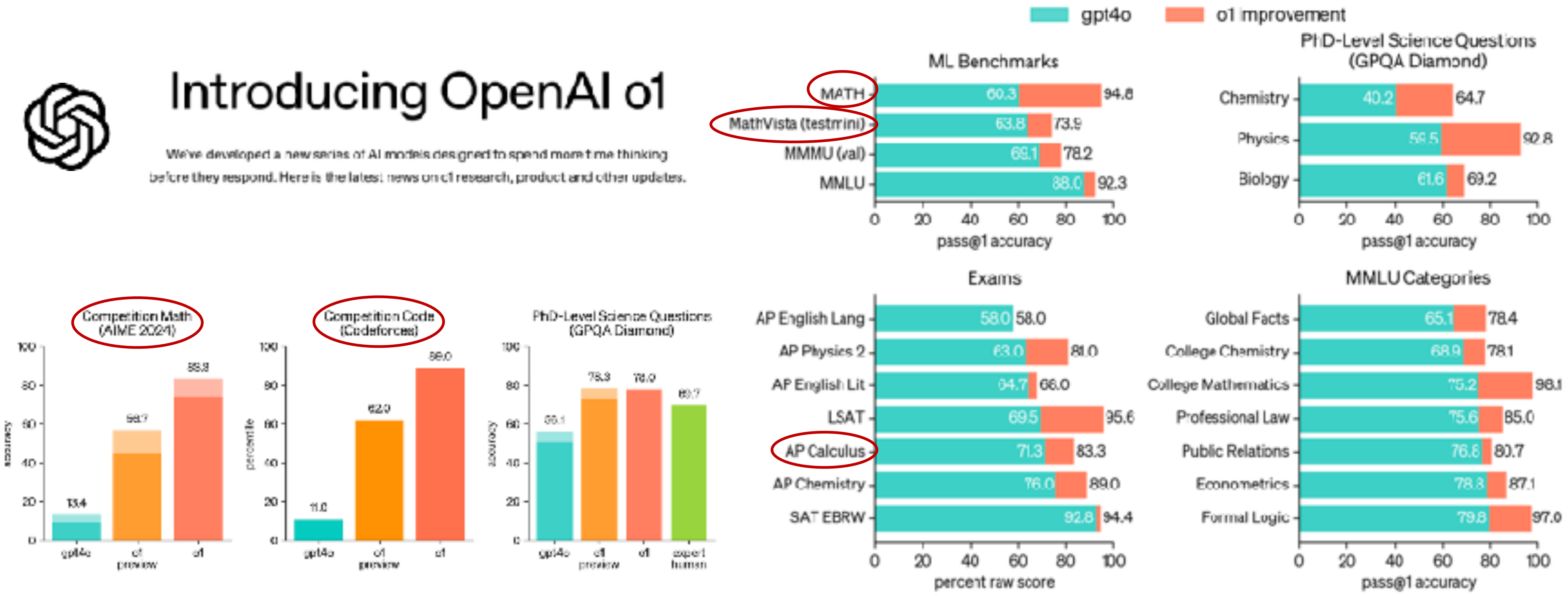
Formal Reasoning Meets LLMs: Towards AI for Mathematics and Verification

Kaiyu Yang

Research Scientist @ Meta FAIR

AI Arms Race in Math and Coding



AI Arms Race in Math and Coding

Gemini 2.0 Flash Thinking

Best for balancing reasoning and speed

BENCHMARK	GEMINI 1.5 PRO 002	GEMINI 2.0 FLASH EXP	GEMINI 2.0 FLASH THINKING EXP 01-21
AIME 2024 (Math)	19.3%	35.5%	73.3%
GPQA Diamond (Science)	57.6%	58.6%	74.2%
MMMU (Multimodal reasoning)	64.9%	70.7%	75.4%

Grok 3 Beta — The Age of Reasoning Agents

AI Arms Race in Math and Coding

[XTX] MARKETS

AI|MO
ARTIFICIAL INTELLIGENCE MATHEMATICAL OLYMPIAD

$10mn AI Mathematical Olympiad Prize Launches

AI achieves silver-medal standard solving International Mathematical Olympiad problems

25 JULY 2024

AlphaProof and AlphaGeometry teams

Google DeepMind

AI Arms Race in Math and Coding

[XTX] MARKETS

AI|MO
ARTIFICIAL INTELLIGENCE MATHEMATICAL OLYMPIAD

$10mn AI Mathematical Olympiad Prize Launches

AI achieves silver-medal standard solving International Mathematical Olympiad problems

25 JULY 2024

AlphaProof and AlphaGeometry teams

Google DeepMind

OpenAI o3-mini

Pushing the frontier of cost-effective reasoning.

EPOCH AI

FrontierMath

A math benchmark testing the limits of AI

	Pass@1	Pass@4	Pass@8
o3-mini (high)	9.2%	16.6%	20.0%
o1-mini	5.8%	9.9%	12.8%
o1	5.5%	10%	12.8%

Why Math and Coding?

- Proxies for **complex reasoning** and **planning**
  - Important in human intelligence; challenging for LLMs
  - Unlimited applications: travel planning, calendar scheduling, etc.

- *Relatively* easy to evaluate
  - Math: check the answers
  - Coding: run unit tests
  - Writing a crime fiction? Composing a symphony?

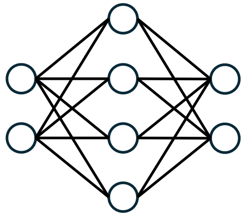
How LLMs are Trained to Solve Math Problems?

- **Supervised finetuning (SFT)**: “Good data is all you need!”

- **Reinforcement learning (RL)**: “Verifiability is all you need!”

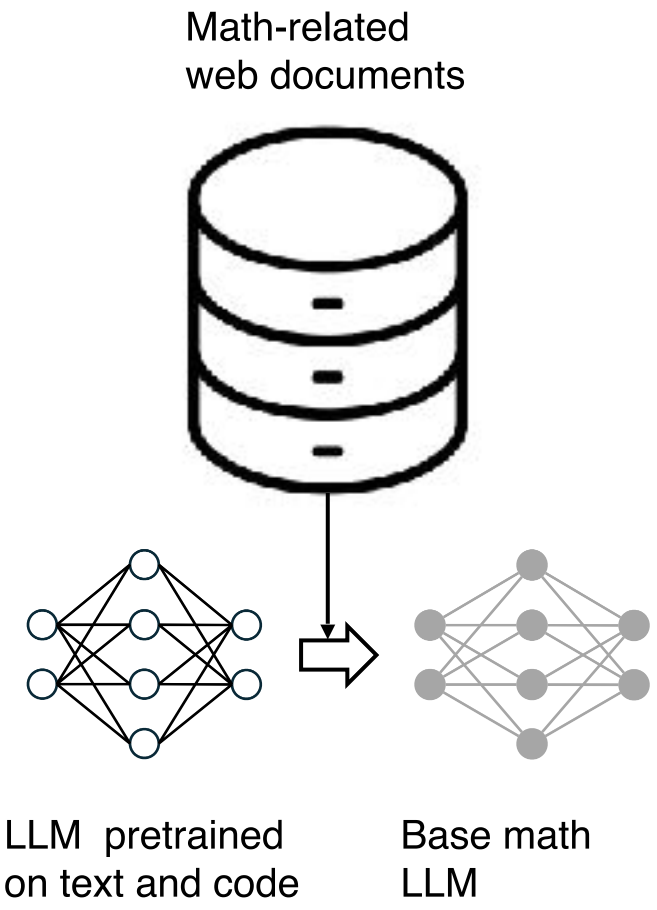
- Methods are straightforward, but the devil is in the details, e.g., data curation/cleaning, infrastructures for training and inference

Supervised Finetuning on Mathematical Data

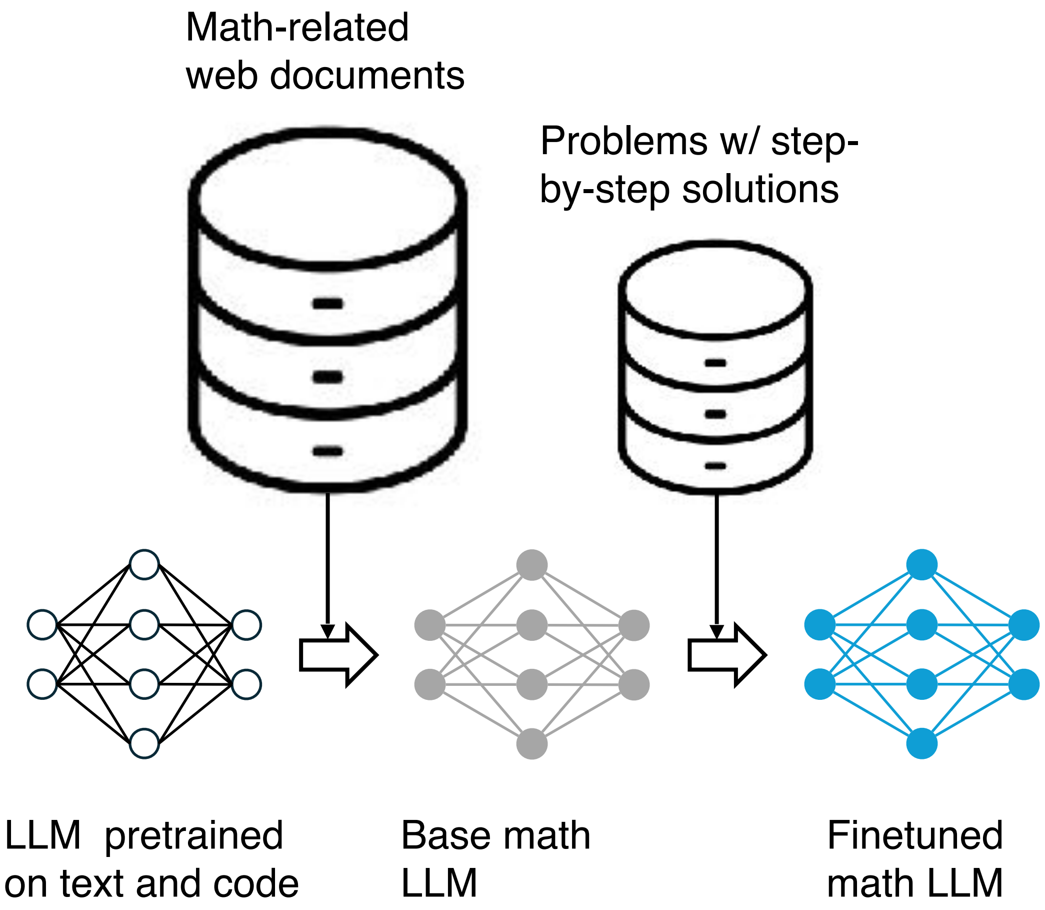


LLM pretrained
on text and code

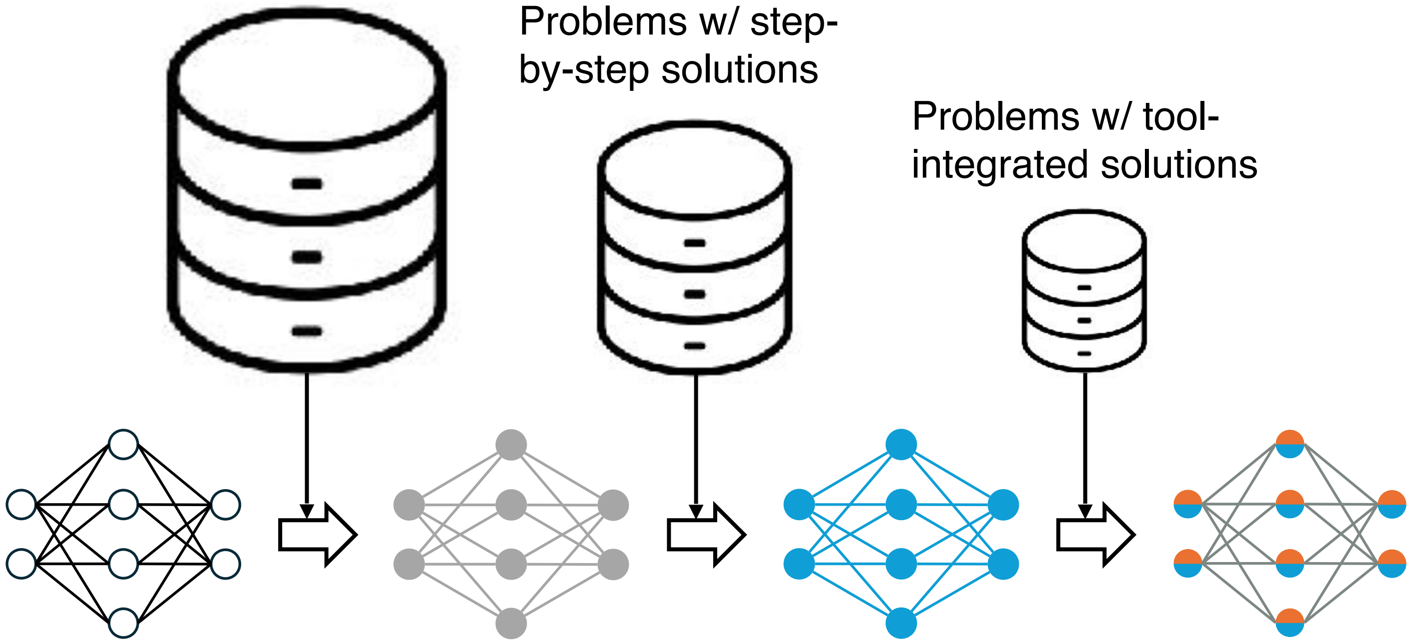
Supervised Finetuning on Mathematical Data



Supervised Finetuning on Mathematical Data



Supervised Finetuning on Mathematical Data



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.

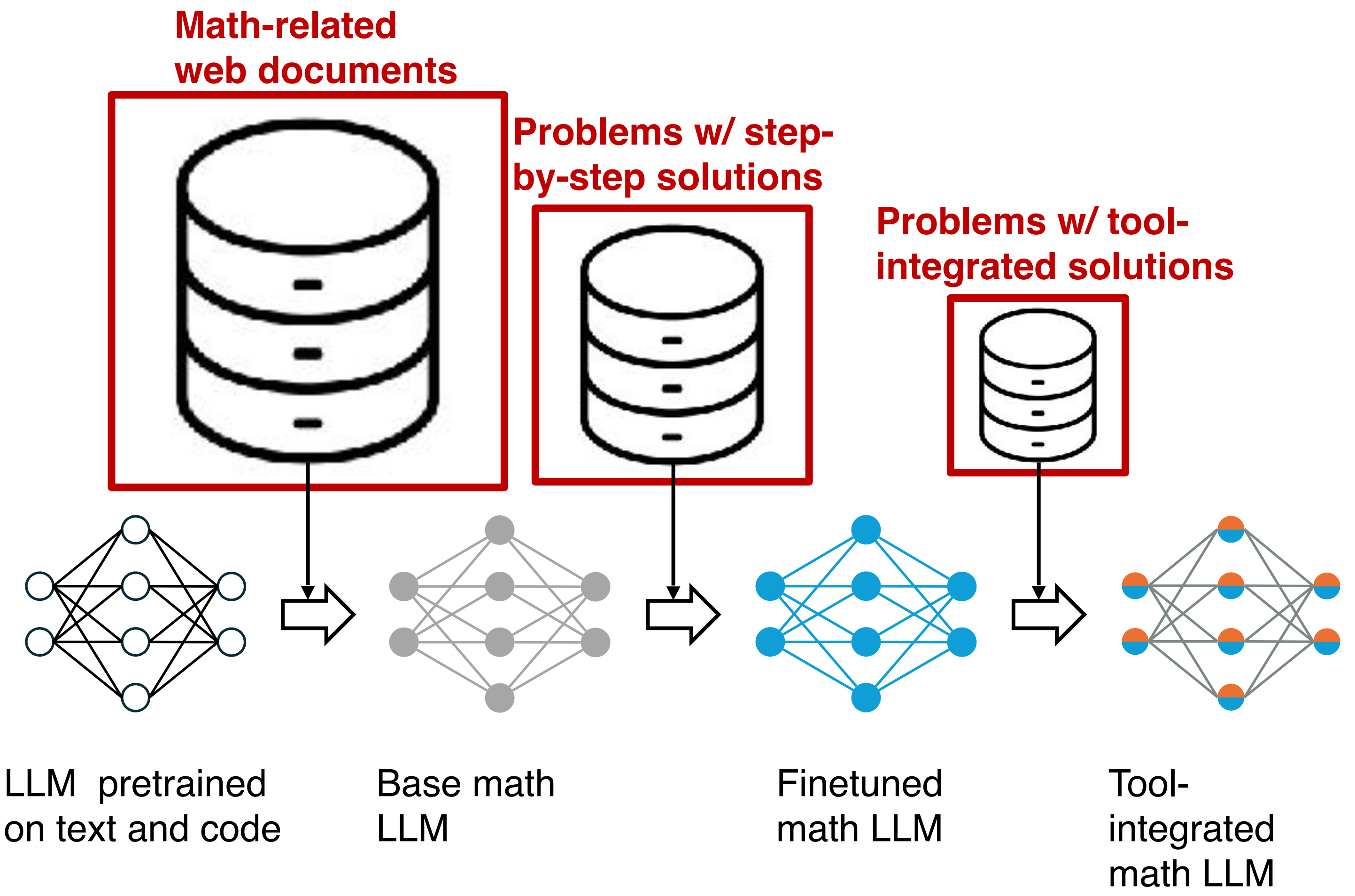
Solution: Let's use `sympy` to calculate and print all possible values for $x + y$.

```
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**



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

- Training data is foremost important
  - Problems + (step-by-step, tool-integrated) solutions curated by humans and LLMs
  - Size of largest public datasets: ~900K

[Li et al., NuminaMath-1.5]

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

- Training data is foremost important
  - Problems + (step-by-step, tool-integrated) solutions curated by humans and LLMs
  - Size of largest public datasets: ~900K

[Li et al., NuminaMath-1.5]

- **What if the data has final answers but not intermediate steps ?**

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**}

Reinforcement Learning on *Verifiable* Problems

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.

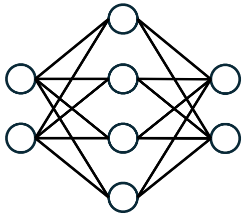
Solution: Let's use `sympy` to calculate and print all possible values for $x + y$.

```
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**}

Reinforcement Learning on *Verifiable* Problems



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 $x + y$.

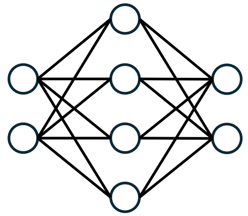
```
def possible_values():
    x, y = symbols("x y")
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve([eq1, eq2], (x, y))
    return [simplify(sol[0] + sol[1]) for sol in solutions]

print(possible_values())
```

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

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

Reinforcement Learning on *Verifiable* Problems

Solution: ... \boxed{**-5, 1, 4**}



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 $x + y$.

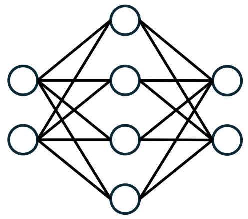
```
def possible_values():
    x, y = symbols("x y")
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve([eq1, eq2], (x, y))
    return [simplify(sol[0] + sol[1]) for sol in solutions]

print(possible_values())
```

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

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

Reinforcement Learning on *Verifiable* Problems

Solution: ... \boxed{**-5, 1, 4**}



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]

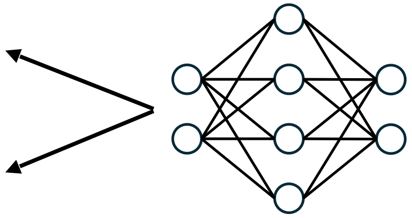
Removing duplicates, the possible values for $x + y$ are \boxed{**-5, 1, 4**}

- Verify the model's solution by comparing the **final answer** with the ground truth

Reinforcement Learning on *Verifiable* Problems

Solution: ... \boxed{-5, 1, 4}

Solution: …\boxed{-2, 6}



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 $x + y$.

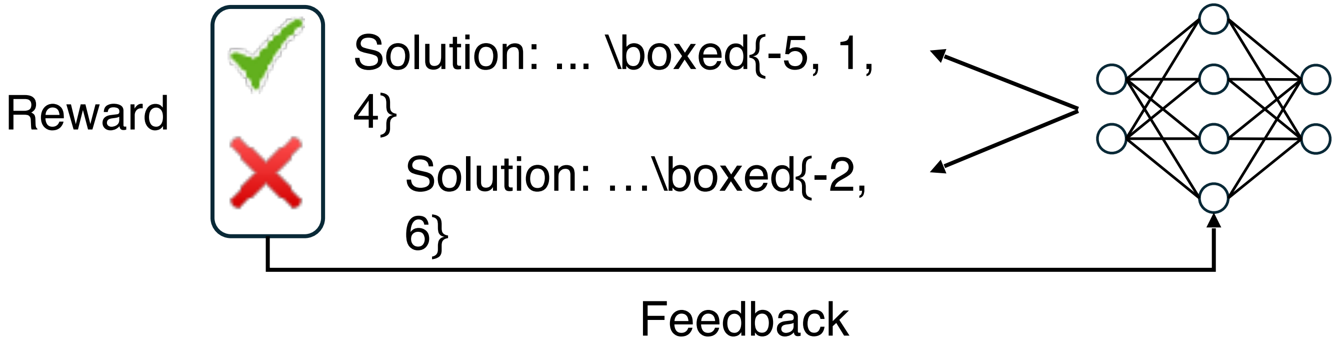
```
def possible_values():
    x, y = symbols("x y")
    eq1 = Eq(x**2 + y**2, 7)
    eq2 = Eq(x**3 + y**3, 10)
    solutions = solve((eq1, eq2), (x, y))
    return [simplify(sol[0] + sol[1]) for sol in solutions]

print(possible_values())
```

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

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

- Verify the model's solution by comparing the **final answer** with the ground truth

Reinforcement Learning on *Verifiable* Problems



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**}

- Verify the model's solution by comparing the **final answer** with the ground truth
- RL algorithms such as GRPO optimize the model to achieve high rewards
  - Popularized by DeepSeek-R1 [Guo et al., 2025]

Reinforcement Learning on *Verifiable* Problems

ACM A.M. Turing Award Honors Two Researchers Who Led the Development of Cornerstone AI Technology

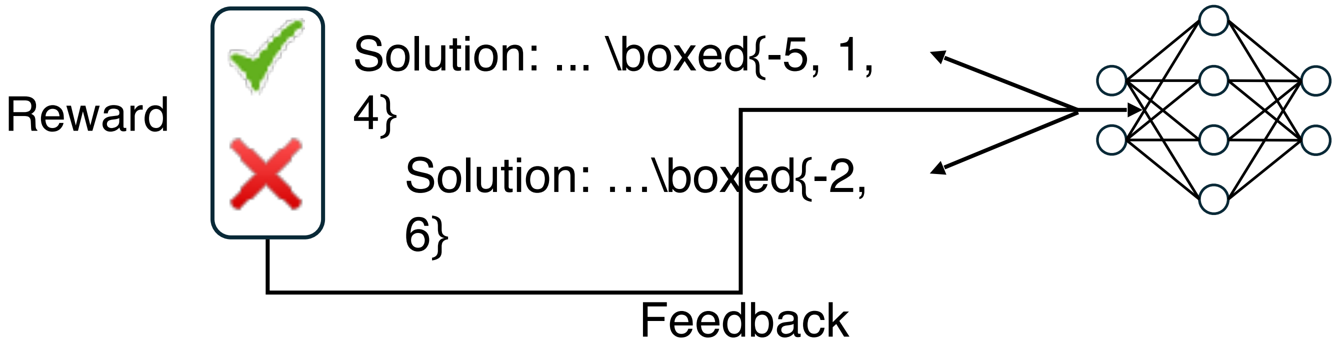
Andrew Barto and Richard Sutton Recognized as Pioneers of Reinforcement Learning

New York, NY, March 5, 2025

Dr. Richard Sutton

Andrew Barto

Reinforcement Learning on *Verifiable* Problems



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**}

- Verify the model's solution by comparing the **final answer** with the ground truth
- RL algorithms such as GRPO optimize the model to achieve high rewards
  - Popularized by DeepSeek-R1 [Guo et al., 2025]
- **The solution must be verifiable, e.g., w/ numeric answers. Not applicable to proofs?**

How LLMs are Trained to Solve Math Problems?

- State-of-the-art math LLM ≈ strong pretrained model + two post-training techniques + marvelous engineering

  - **Supervised finetuning (SFT)**: “Good **data** is all you need!”

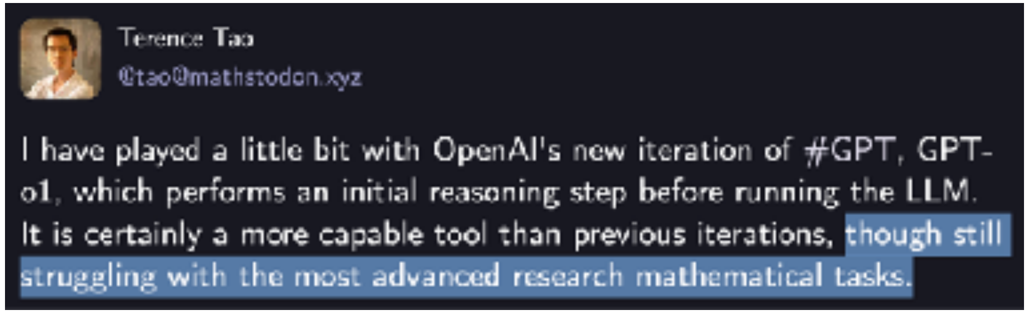
  - **Reinforcement learning (RL)**: “**Verifiability** is all you need!”

How LLMs are Trained to Solve Math Problems?

- State-of-the-art math LLM ≈ strong pretrained model + two post-training techniques + marvelous engineering
    - **Supervised finetuning (SFT)**: “Good **data** is all you need!”
    - **Reinforcement learning (RL)**: “**Verifiability** is all you need!”
- **Will AI soon “solve mathematics”?**

Gap 1: Pre-college Math -> Advanced Math

- Existing successes are mostly on pre-college math, e.g., AIME, IMO
- LLMs struggle with more advanced math, e.g., mathematical research



EPOCH AI

- o3's FrontierMath results come with caveats
- How to solve problems w/o numeric answers?

Gap 2: Guessing Answers -> Writing Proofs

- LLMs struggle to generate valid proofs

William Lowell PUTNAM Mathematical Competition

to be quite a big one. For example I was very unimpressed by the ChatGPT answers to the recent Putnam exam posted here — as far as I can see only question B4 was answered adequately by the machine, most other answers are worth one or two out of 10 at most. So I was expecting this dataset to remain pretty unattackable for a couple of years.

[Kevin Buzzard “Can AI do maths yet? Thoughts from a mathematician” 2024]

Gap 2: Guessing Answers -> Writing Proofs

- LLMs struggle to generate valid proofs

to be quite a big one. For example I was very unimpressed by the ChatGPT answers to the recent Putnam exam posted here — as far as I can see only question B4 was answered adequately by the machine, most other answers are worth one or two out of 10 at most. So I was expecting this dataset to remain pretty unattackable for a couple of years.

[Kevin Buzzard “Can AI do maths yet? Thoughts from a mathematician” 2024]

tors, we evaluated several state-of-the-art reasoning models on the six problems from the 2025 USAMO within hours of their release. Our results reveal that all tested models struggled significantly, achieving less than 5% on average. Through

[Petrov et al. "Proof or Bluff? Evaluating LLMs on 2025 USA Math Olympiad" 2025]

Gap 2: Guessing Answers -> Writing Proofs

- LLMs struggle to generate valid proofs

We will show that if

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

then

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

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

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

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

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

Thus,

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

LLMs Alone are Not Enough

- Current math LLMs rely heavily on data and verifiability

- Data scarcity
  - Limited to data-rich domains, e.g., pre-college math
  - Cannot tackle advanced math or proofs
- Lack of verifiability
  - Solutions can only be evaluated by comparing with the ground truth
  - Limited to problems with numeric solutions, e.g., GSM8K, MATH
  - Not applicable to most problems in advanced math

Formal Mathematical Reasoning

- Our position paper
- Mathematical reasoning grounded in *formal systems*, e.g.,
  - First/higher-order logic
  - Dependent type theory
  - Computer programs & formal specifications
- Formal environments can verify proofs and provide automatic feedback
  - Verification enables rigorous evaluation of reasoning
  - Learning from feedback mitigates data scarcity
- Integrating formal reasoning and LLMs' informal reasoning

The Missing Ingredient: Formal Reasoning

Formal Mathematical Reasoning:
A New Frontier in AI

Kaiyu Yang[1], Gabriel Poesia[2], Jingxuan He[3],
Wenda Li[4], Kristin Lauter[1], Swarat Chaudhuri[5], Dawn Song[3]
[1]Meta FAIR, [2]Stanford University, [3]UC Berkeley, [4]University of Edinburgh, [5]UT Austin

[Yang et al. "Formal Mathematical Reasoning: A New Frontier in AI" 2024]

- Mathematical reasoning grounded in *formal systems*, e.g.,
  - First/higher-order logic, dependent type theory
  - Computer programs & formal specifications
- Formal systems can verify proofs and provide automatic feedback
  - Learning from feedback mitigates data scarcity
  - Verification enables rigorous evaluation of reasoning
- We need to integrate formal reasoning with informal reasoning by LLMs

Proof Assistants (Interactive Theorem Provers)

- Programming languages for writing formal math and software

[Nipkow et al., **Isabelle**, 2002] [Barras et al., **Coq**, 1997] [de Moura et al., **Lean**, 2015]

```
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat
```

Define natural numbers

```
def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')
```

Define addition

```
theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]

theorem add_succ (m n : Nat) : add (.succ m) n = .succ (add m n) := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]

theorem add_comm (m n : Nat) : add m n = add n m := by
  induction n <;> simp [add, add_zero]
  simp [*, add_succ]
```

State and prove theorems about natural number addition

e.g., $a + b = b + a$

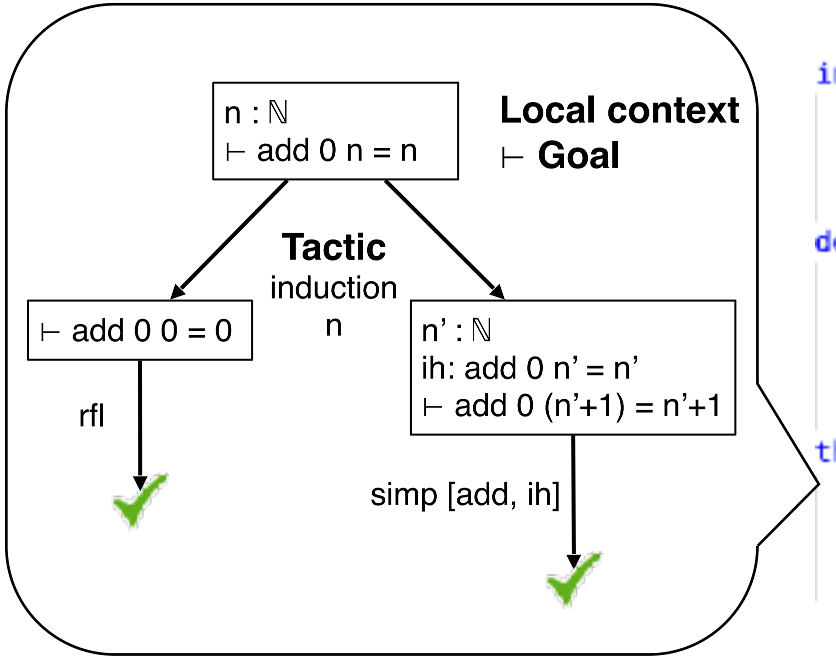
Formalizing Mathematics in Lean

Lean file

```
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat

def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')

theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]
```

Formalizing Mathematics in Lean

Proof tree

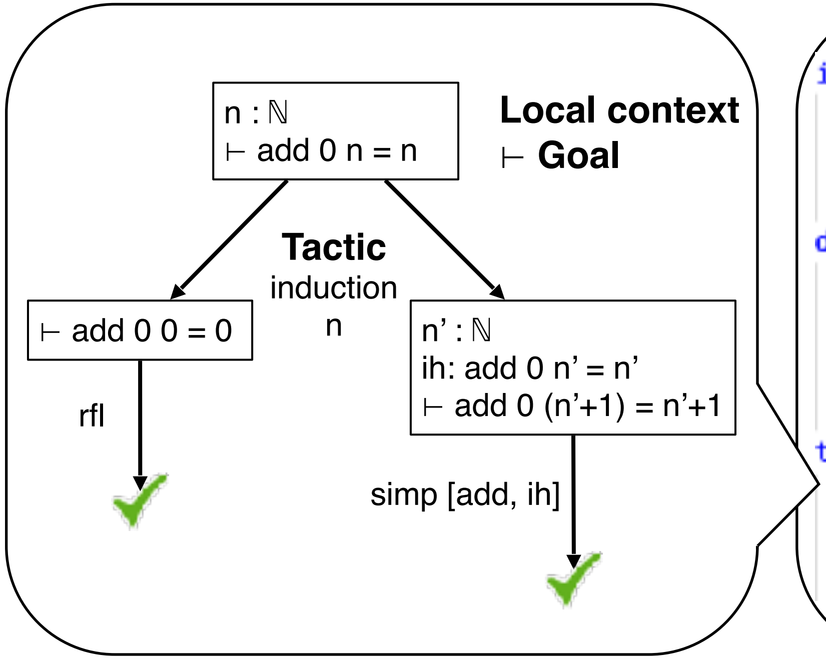


Lean file

```
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat

def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')

theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]
```

Formalizing Mathematics in Lean

Proof tree



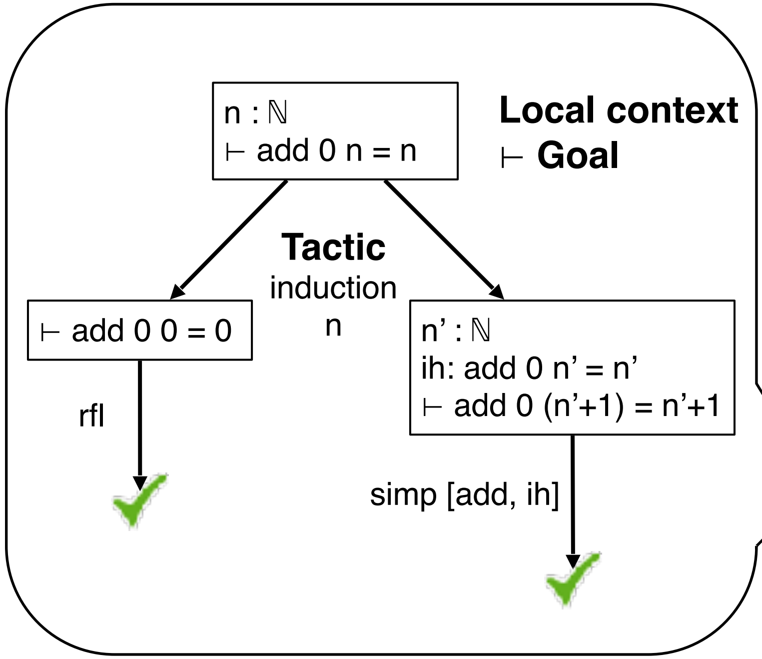
Lean file

```
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat

def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')

theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]
```

Project

Formalizing Mathematics in Lean

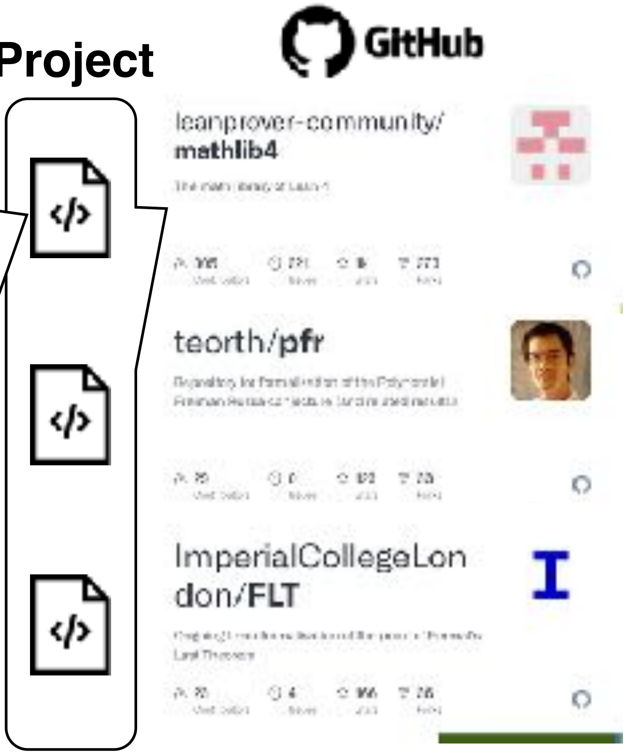
Proof tree



Lean file

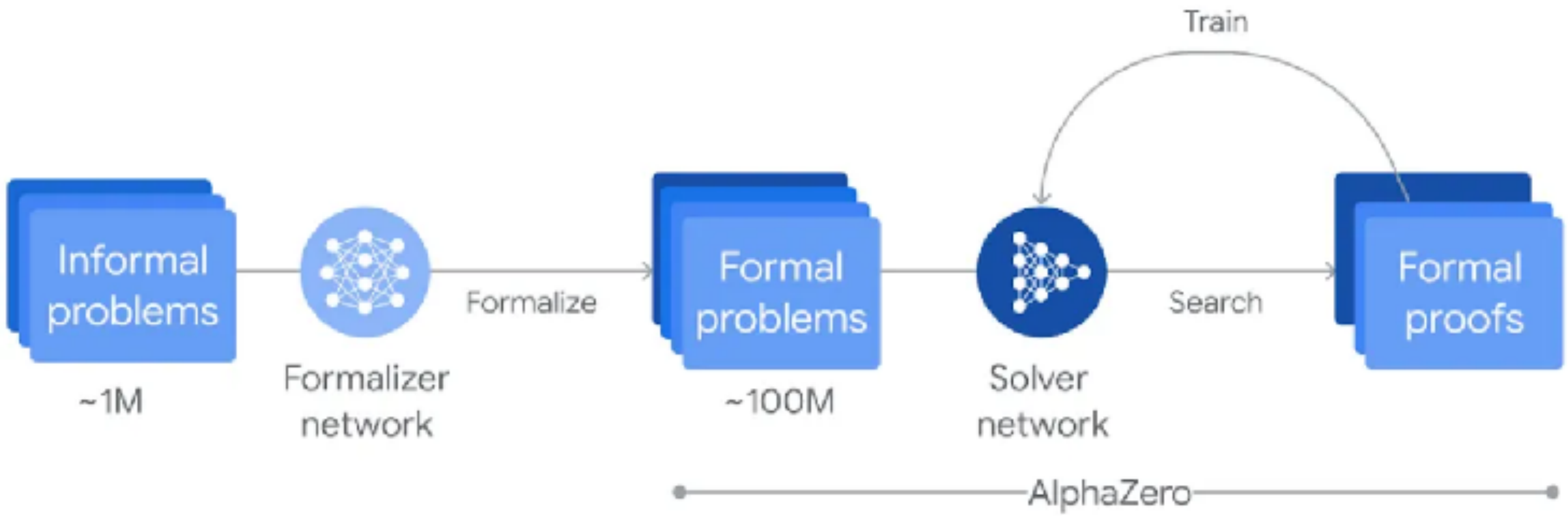
```lean
inductive Nat where
  | zero : Nat
  | succ : Nat → Nat

def add (m n : Nat) : Nat :=
  match n with
  | .zero => m
  | .succ n' => .succ (add m n')

theorem add_zero (n : Nat) : add .zero n = n := by
  induction n with
  | zero => rfl
  | succ n ih => simp [add, ih]
```

Project



Example of AI + Lean: AlphaProof

- Large-scale search and reinforcement learning using feedback from Lean



[Google DeepMind "AI achieves silver-medal standard solving International Mathematical Olympiad problems" 2024]

AI Meets Formal Mathematics

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

AI Meets Formal Mathematics

- Theorems and proofs are represented formally in Lean
- Lean can check if the proof is correct. No room for hallucination

```
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⟩
```

AI Meets Formal Mathematics

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n! + 1$. We can derive $p > n$ by noting that $n! + 1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. □

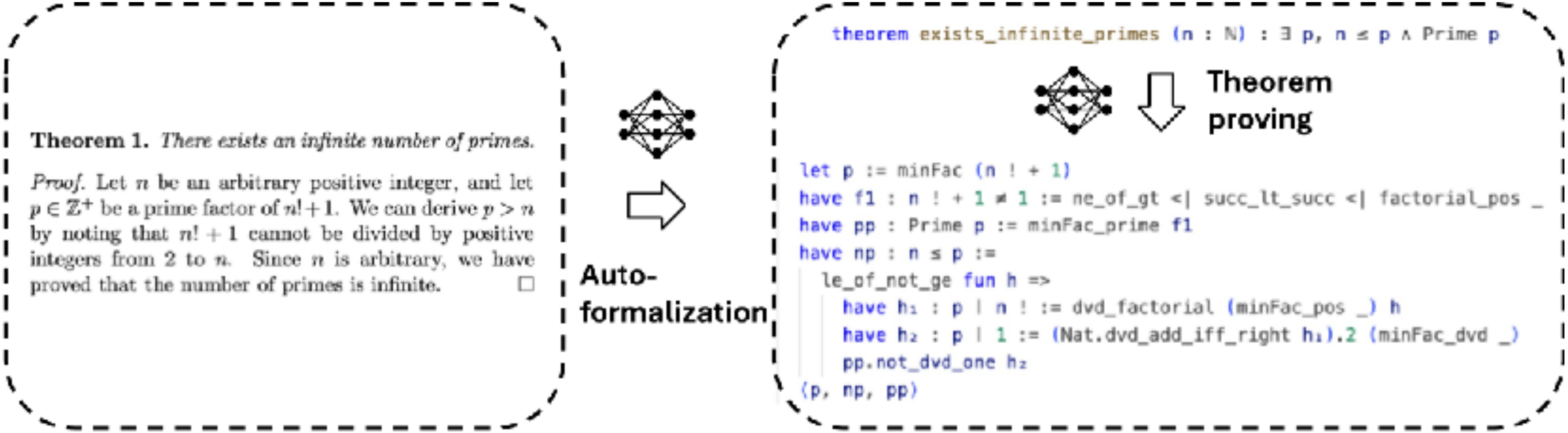
Informal math

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


```
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⟩
```

Formal theorem statement and proof

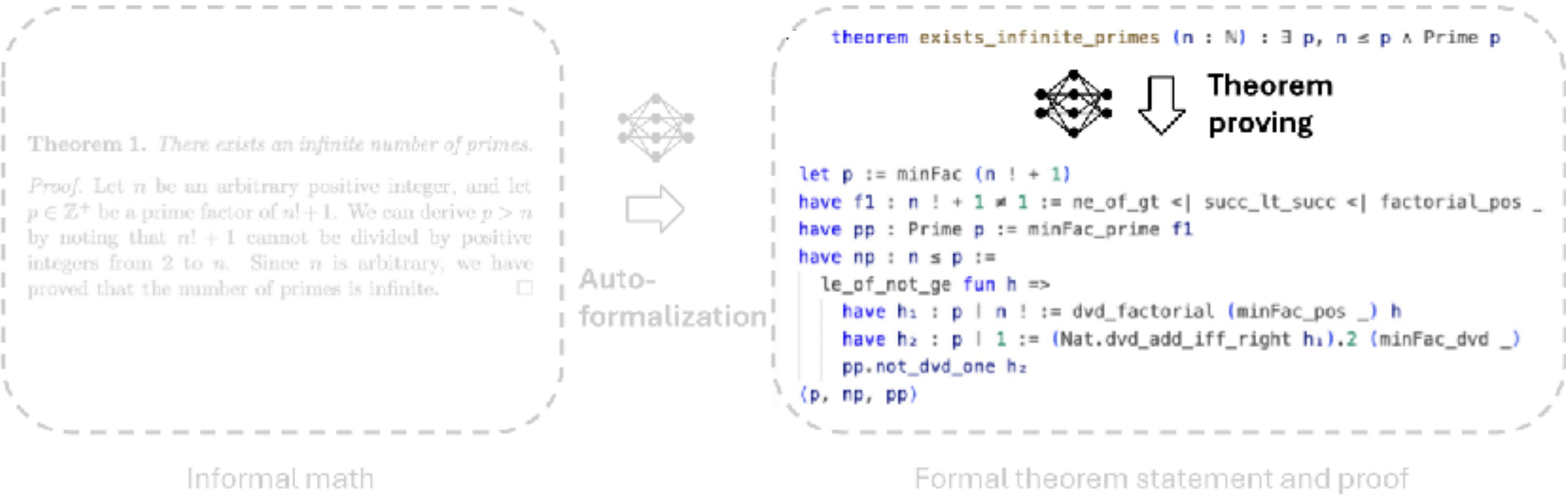
AI Meets Formal Mathematics



Informal math

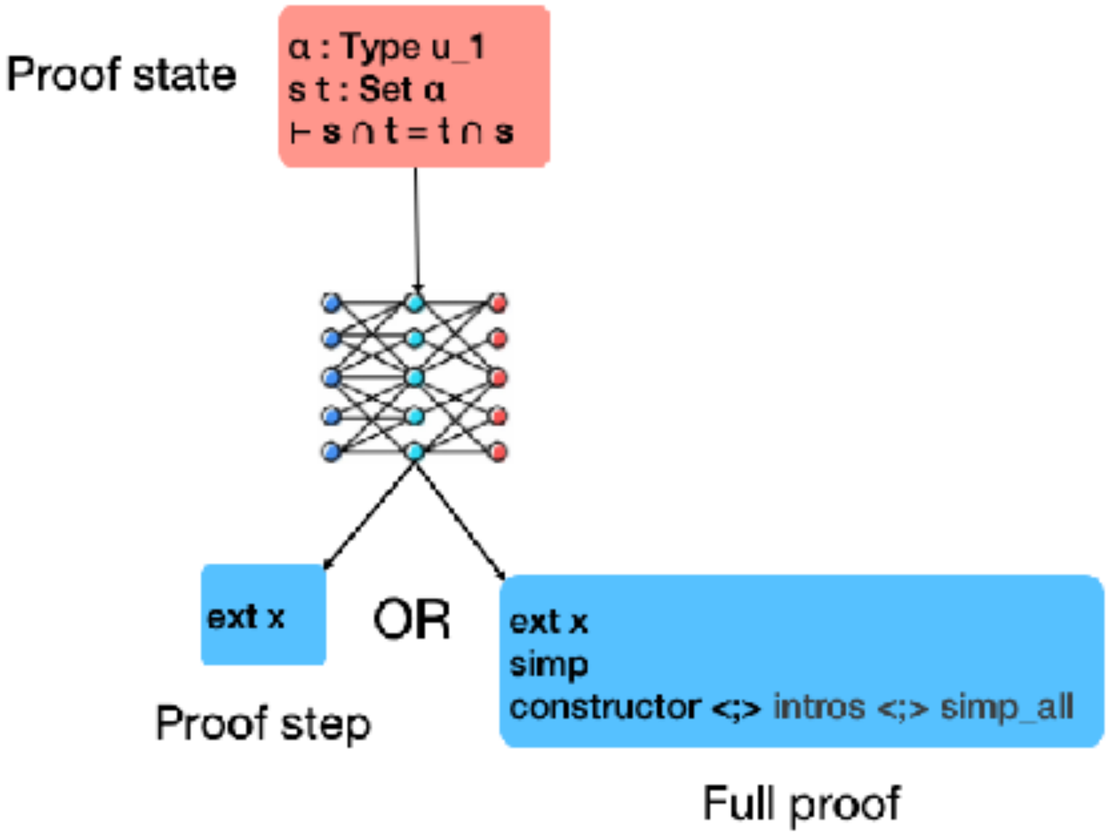
Formal theorem statement and proof

AI Meets Formal Mathematics



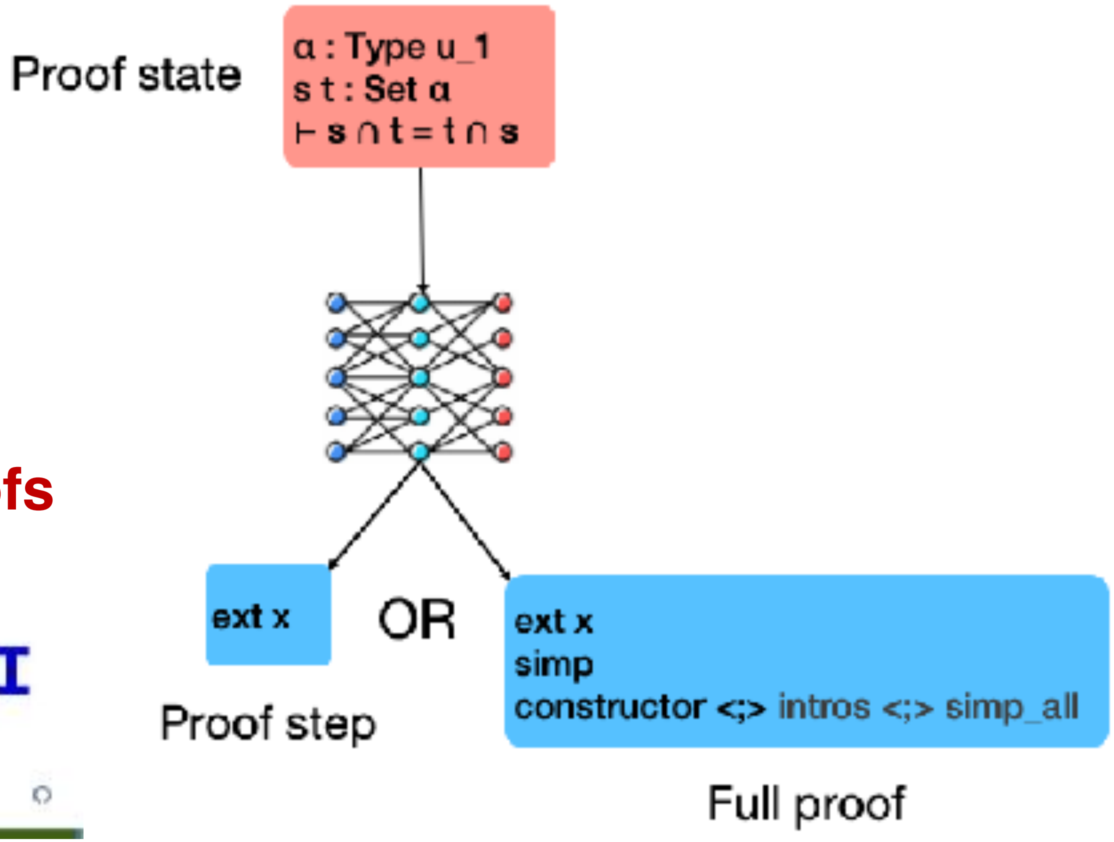
LLMs for Theorem Proving

- We can train LLMs to generate either
    - Next steps in the proof (a.k.a. tactic)
    - Complete proofs
- Proof steps can be assembled into complete proofs using search algorithms
- How to generate the next step?



LLMs for Theorem Proving

- We can train LLMs to generate either
  - Next steps in the proof (a.k.a. tactic)
  - Complete proofs
- Proof steps can be assembled into complete proofs using search algorithms
- How to generate the next step?
  - **Learn from human-written formal proofs**



Machine Learning for Predicting the Next Step

- Classical ML algorithms, e.g., KNN

[Gauthier et al. "TacticToe: Learning to Prove with Tactics" 2018]

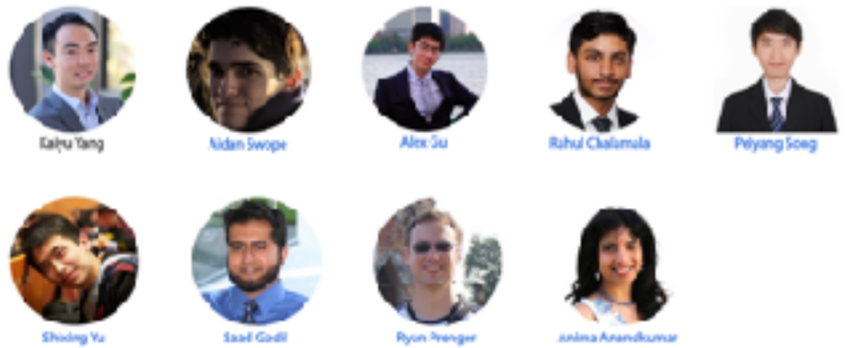
- Deep neural networks

[Huang et al. “GamePad: A Learning Environment for Theorem Proving” ICLR 2019]
[Yang et al. “Learning to Prove Theorems via Interacting with Proof Assistants” ICML 2019]
[Yang et al. “Learning to Prove Theorems via Interacting with Proof Assistants” ICML 2019]
[Bansal et al. "HOList: An Environment for Machine Learning of Higher-Order Theorem Proving" ICML 2019]

- LLMs

[Polu and Sutskever "Generative Language Modeling for Automated Theorem Proving" 2020]
[Lample et al. "HyperTree Proof Search for Neural Theorem Proving" NeurIPS 2022]
[Han et al. "Proof Artifact Co-training for Theorem Proving with Language Models" ICLR 2022]
…

LeanDojo



LeanDojo: Theorem Proving with Retrieval-Augmented Language Models

Kaiyu Yang[1], Aidan M. Swope[2], Alex Gu[3], Rahul Chalamala[1], Peiyang Song[4], Shixing Yu[5], Saad Godil[2], Ryan Prenger[2], Anima Anandkumar[1,2]
[1]Caltech, [2]NVIDIA, [3]MIT, [4]UC Santa Barbara, [5]UT Austin
https://leandojo.org

[Yang et al. "LeanDojo: Theorem Proving in Lean using Language Models" NeurIPS 2023]

- Previous LLM-based provers are private
- LeanDojo provides open-source
  - Data for training and evaluation
  - Trained model checkpoints
  - Tools for extracting data and interacting with Lean

LeanDojo

Data extraction

LeanDojo Benchmark

- 98,641 theorems and proofs
- 217,639 tactics
- 129,162 premises

Lean

LeanDojo

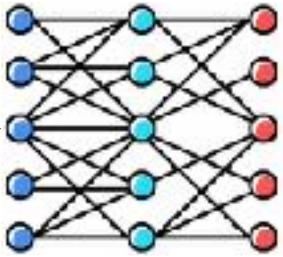
LEAN

Data extraction →

LeanDojo Benchmark

- 98,641 theorems and proofs
- 217,639 tactics
- 129,162 premises

Training →



Lean

Machine learning model

LeanDojo

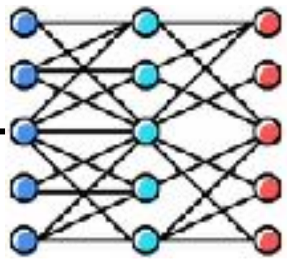
Prove theorems by Interaction

Data extraction

LeanDojo Benchmark

- 98,641 theorems and proofs
- 217,639 tactics
- 129,162 premises

Training

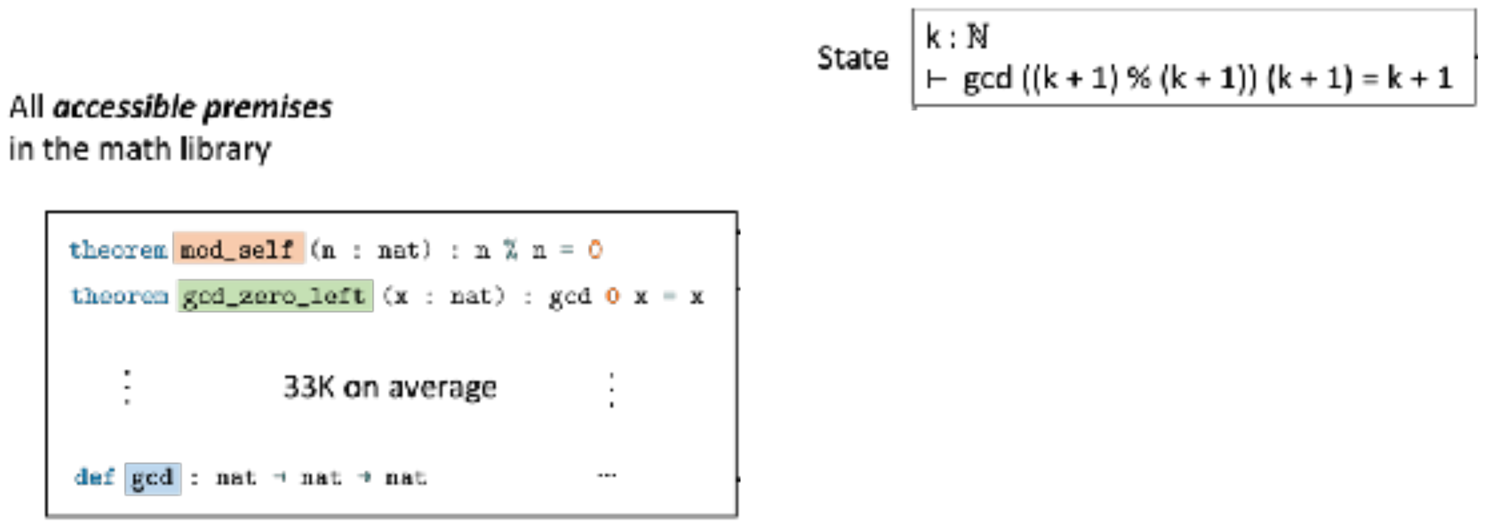


Lean

Machine learning model

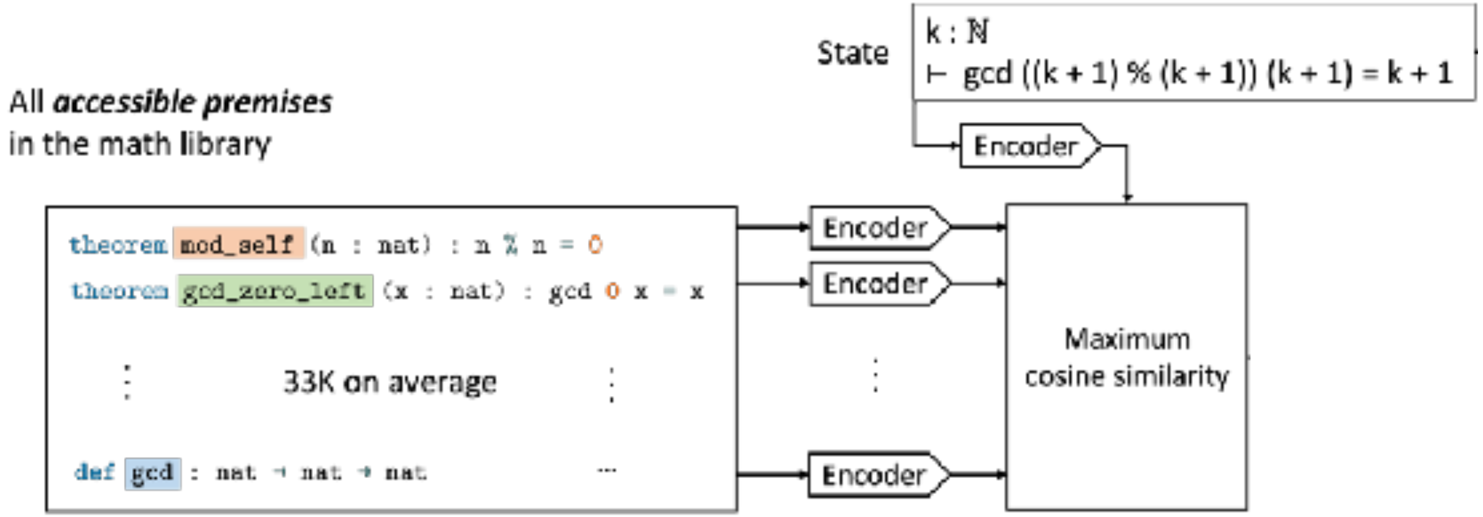
Retrieval-Augmented Prover (ReProver)

- Given a state, we retrieve premises from the set of **all accessible premises**



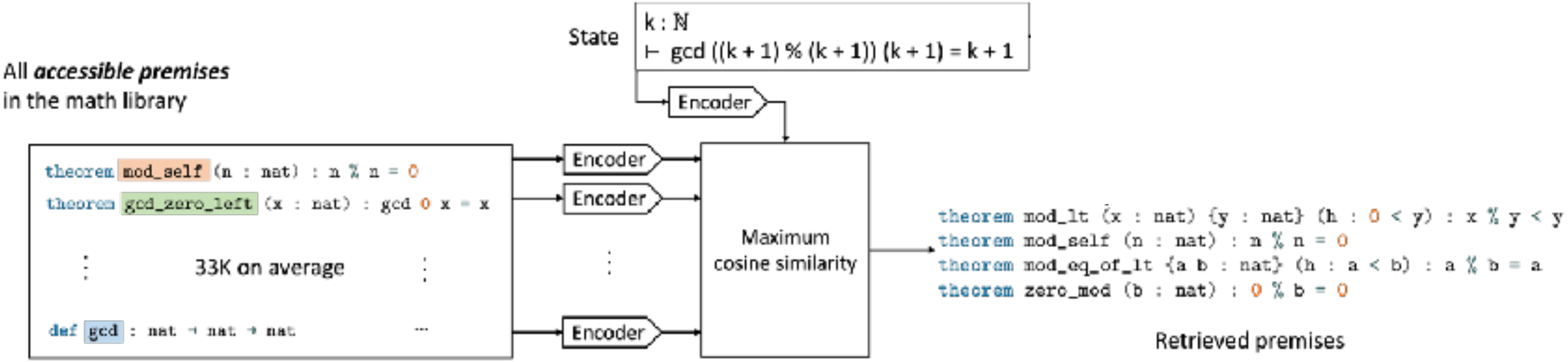
Retrieval-Augmented Prover (ReProver)

- Given a state, we retrieve premises from the set of **all accessible premises**



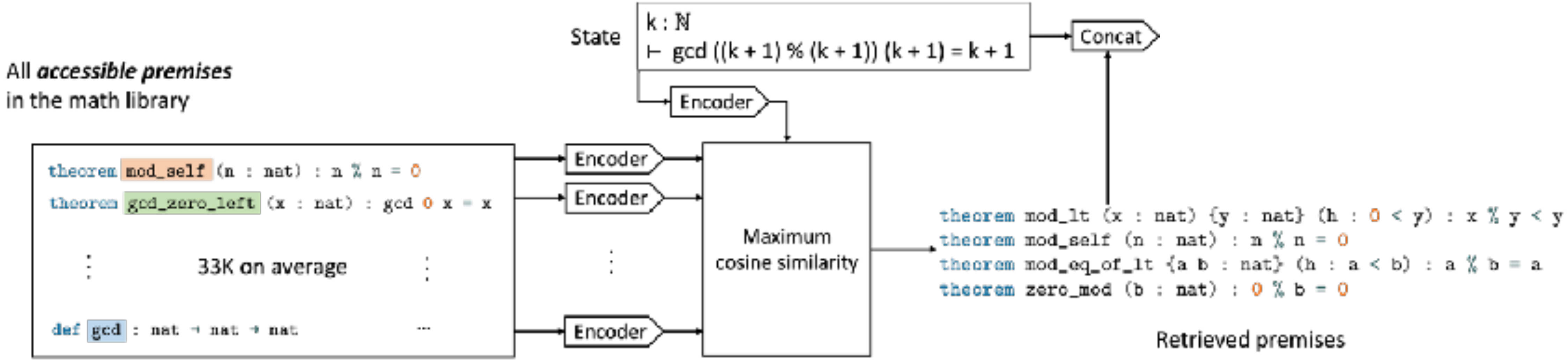
Retrieval-Augmented Prover (ReProver)

- Given a state, we retrieve premises from the set of **all accessible premises**



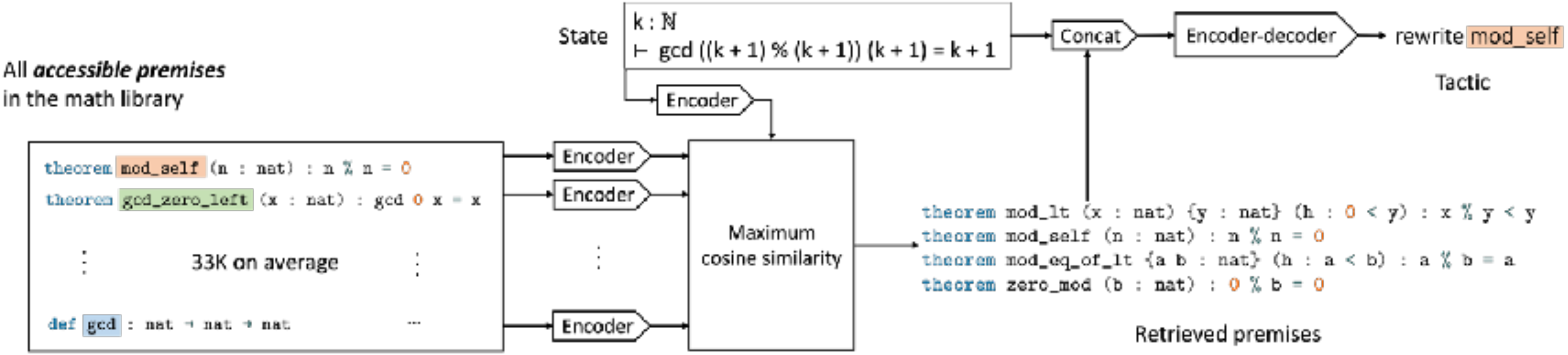
Retrieval-Augmented Prover (ReProver)

- Given a state, we retrieve premises from the set of **all accessible premises**
- Retrieved premises are concatenated with the state and used for tactic generation

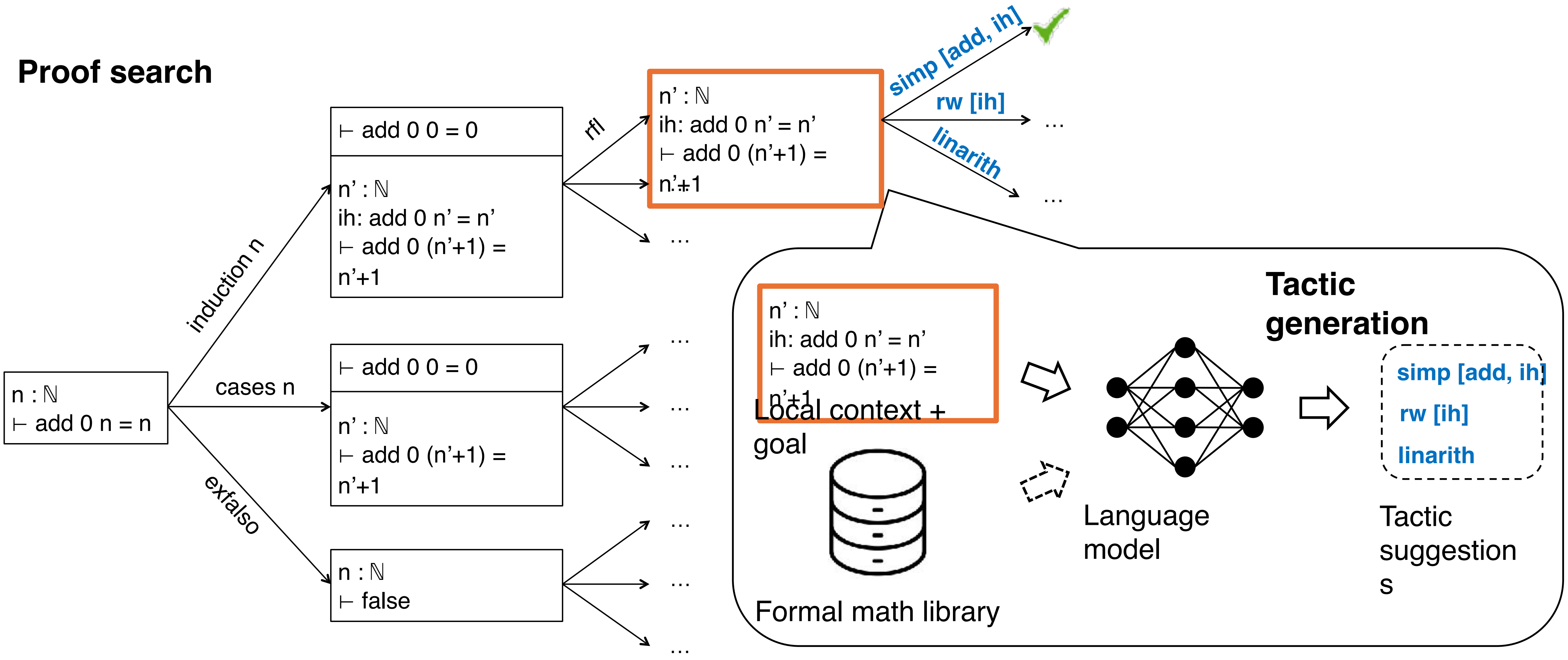


Retrieval-Augmented Prover (ReProver)

- Given a state, we retrieve premises from the set of **all accessible premises**
- Retrieved premises are concatenated with the state and used for tactic generation



Summary: A Typical Neural Theorem Prover



55

Goedel-Prover

A New Frontier in Open-source Automated Theorem Proving

Yong Lin*[1] Shange Tang*[1] Bohan Lyu[2] Jiayun Wu[2]
Hongzhou Lin[3] Kaiyu Yang[4] Jia Li[5] Mengzhou Xia[1]
Danqi Chen[1] Sanjeev Arora[1] Chi Jin[1]

[1]Princeton Language and Intelligence, Princeton University
[2]Tsinghua University, [3]Amazon, [4]Meta FAIR, [5] Numina

arXiv HuggingFace Code

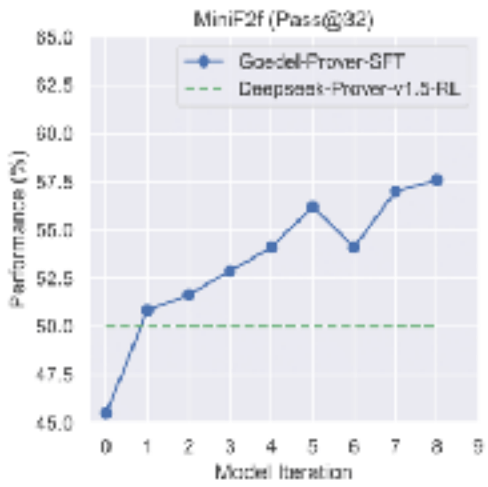


Figure 1: Iterative Performance on miniF2F

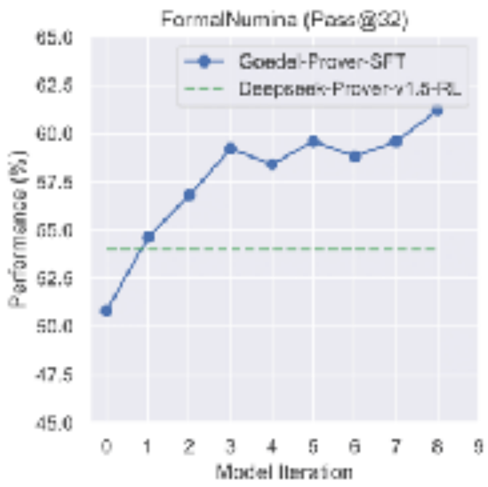


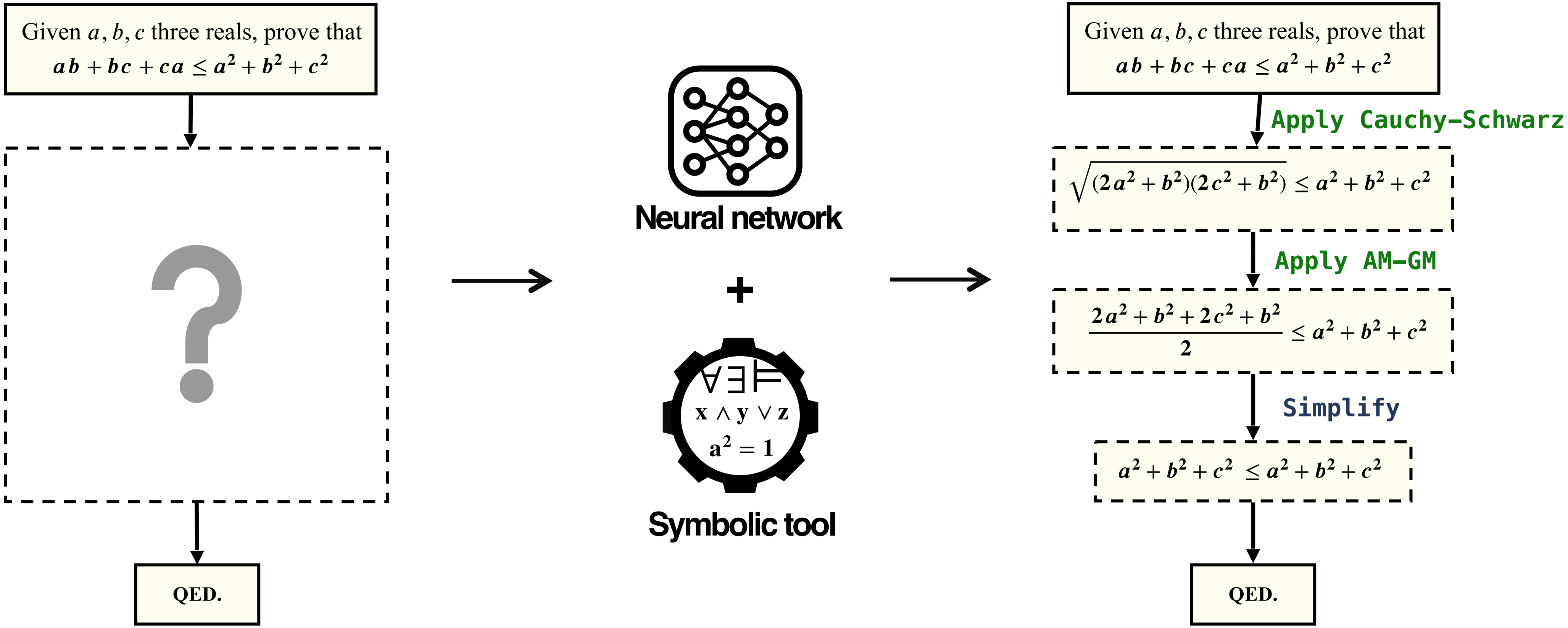
Figure 2: Iterative Performance on FormalNumina

Limitations

- LLMs work well in domains with abundant data, but novel mathematical research is data-scarce
- The “action space” in proving mathematical theorems large
  - Go: 19x19 board. Math: infinite?
  - Hard to cover the space uniformly by human-created data
  - Exploration is difficult in reinforcement learning

Taming the Action Space in Proving Inequalities

[Li et al. "Proving Olympiad Inequalities by Synergizing LLMs and Symbolic Reasoning" ICLR 2025]



Infinite Proof Search Space

Problem: If $a,\ b,\ c$ are positive reals and $a^2 + b^2 + c^2 = 1$, then

$$\frac{1}{a^2+2} + \frac{1}{b^2+2} + \frac{1}{c^2+2} \le \frac{1}{6ab+c^2} + \frac{1}{6bc+a^2} + \frac{1}{6ca+b^2}$$

AM-GM (1)

$$\frac{1}{2\sqrt{2a}} + \frac{1}{2\sqrt{2b}} + \frac{1}{2\sqrt{2c}} \le \frac{1}{6ab+c^2} + \frac{1}{6bc+a^2} + \frac{1}{6ca+b^2}$$

... ...

AM-GM (2)

$$\frac{1}{a^2+2} + \frac{1}{b^2+2} + \frac{1}{c^2+2} \le 3[(6ab+c^2)(6bc+a^2)(6ab+c^2)]^{-\frac{1}{3}}$$

... ...

...

Rewrite

$$\frac{1}{3a^2+2b^2+c^2} + \frac{1}{3b^2+2a^2+2c^2} + \frac{1}{3c^2+2a^2+2b^2} \le \frac{1}{6ab+c^2} + \frac{1}{6bc+a^2} + \frac{1}{6ca+b^2}$$

... ...

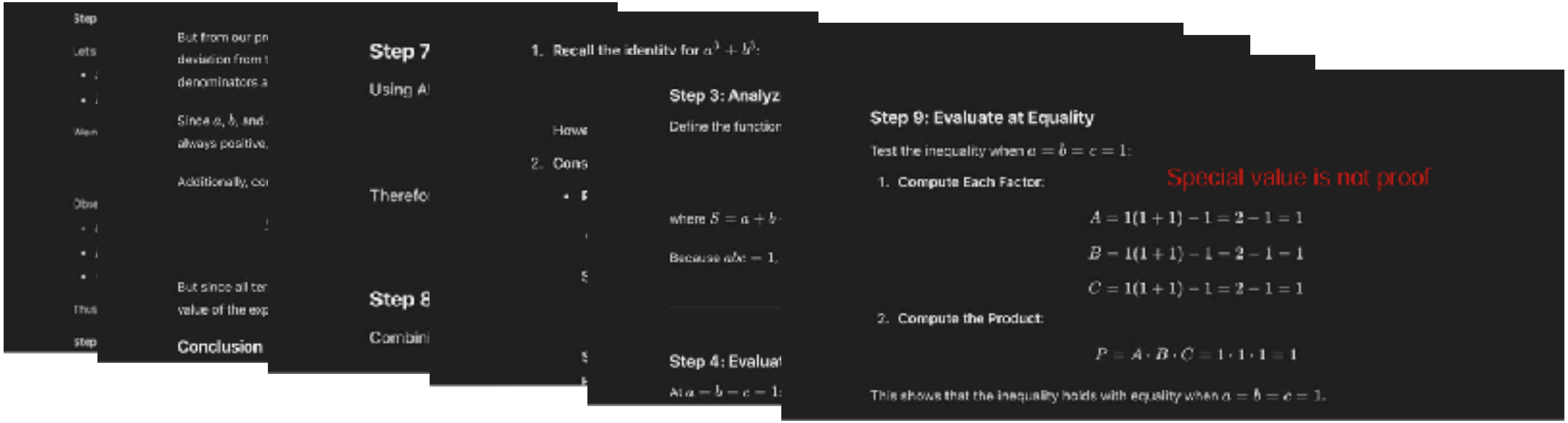
Titu

$$\frac{1}{a^2+2} + \frac{1}{b^2+2} + \frac{1}{c^2+2} \le \frac{9}{6ab+6bc+6ca+a^2+b^2+c^2}$$

... ...

>10,000 potential one-steps options

Manually Checking o1's Proofs



...

	o1-preview	o3-mini	DeepSeek-R1	Gold medalists
#Solved Olympiad-level Inequalities	0/20	3/20	4/20	**15/20**

Tactic Generation & Pruning

- We categorize the steps in inequality proving into two types:
  1) Scaling: substitute the given inequality using a known lemma (e.g., Cauchy-Schwarz)
  2) Rewriting: transform the given inequality into an equivalent form
- **We enumerate and prune the scaling tactics using symbolic tools**

Given a, b, c three reals, prove that

$$\boxed{ab + bc + ca} \leq a^2 + b^2 + c^2$$

$$\boxed{\sqrt{(2a^2 + b^2)(2c^2 + b^2)}} \leq a^2 + b^2 + c^2$$

Pattern match

`Cauchy_Schwarz_left_sqrt_2vars`

$$\boxed{u_1 v_1 + u_2 v_2} \leq \boxed{\sqrt{u_1^2 + u_2^2}\sqrt{v_1^2 + v_2^2}}$$

Enumerate and prune all patterns using symbolic tools

~~u1:=1, u2:=1, v1:=a, v2:=b~~
~~u1:=a, u2:=1, v1:=1, v2:=b~~
u1:=c, u2:=b, v1:=a, v2:=c
~~u1:=1, u2:=1, v1:=a, v2:=b~~
~~u1:=1, u2:=b, v1:=a, v2:=1~~
~~u1:=1, u2:=a, v1:=b, v2:=1~~
… …

Tactic Generation & Pruning

- We categorize the steps in inequality proving into two types:
    1) Scaling: substitute the given inequality using a known lemma (e.g., Cauchy-Schwarz)
    2) Rewriting: transform the given inequality into an equivalent form (e.g., fraction reduction)

Given a, b, c three reals, prove that

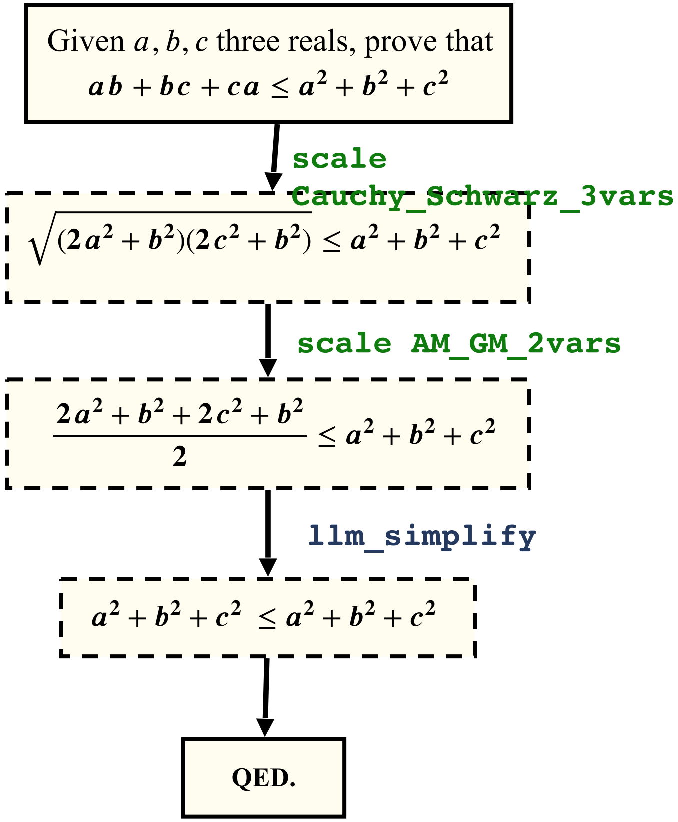
$$\boxed{\boldsymbol{ab + bc + ca \le a^2 + b^2 + c^2}}$$

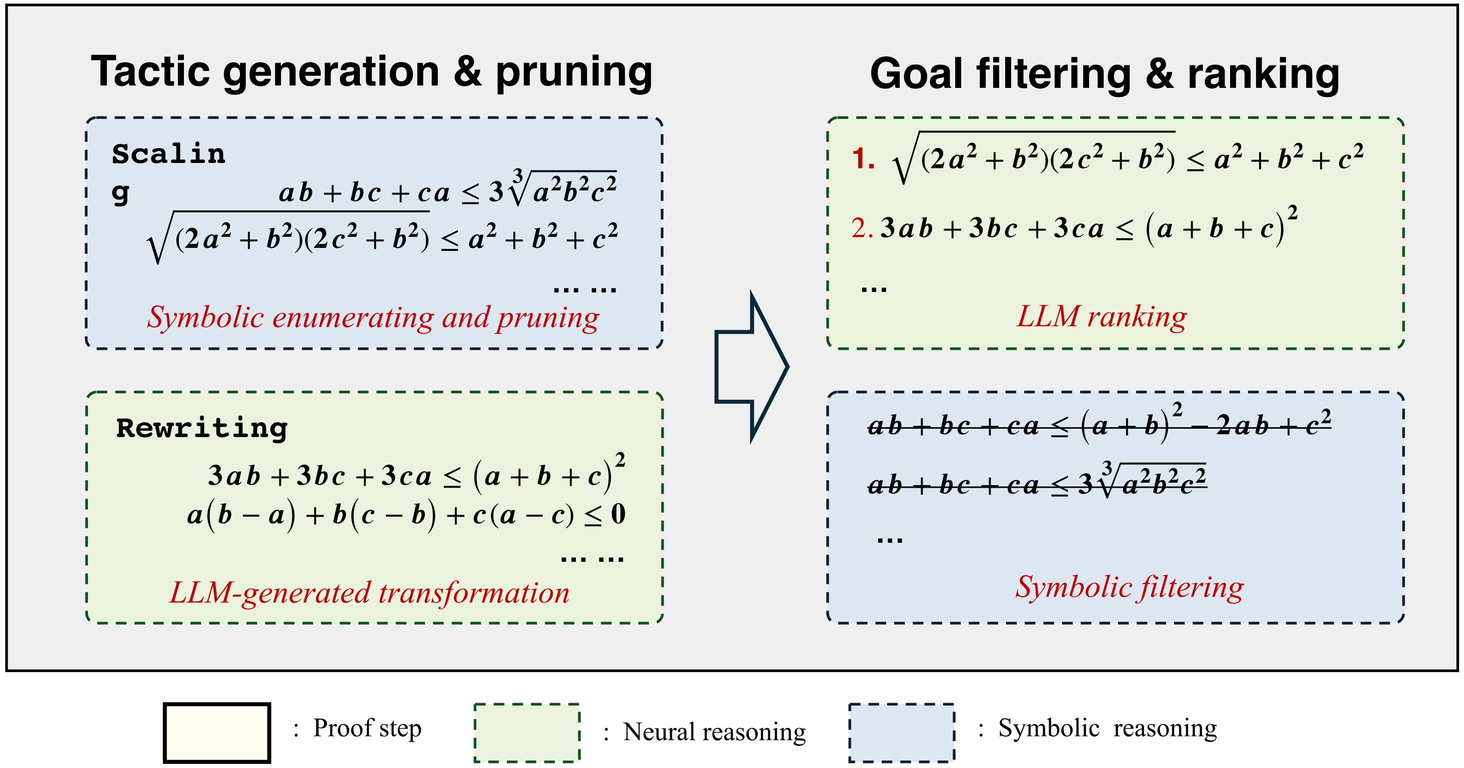
Prompt: You should rewrite the inequality given by the user according to the rule `x*y+y*z = y*(x+z)`
Please reason step by step
~~Problem: {problem}~~

Let's rewrite the inequality \(ab + bc + ca \leq a^2 + b^2 + c^2 \) step by step according to the rule \(xy + yz = y(x + z) \).
Step 1: …
… …
Conclusion:
The inequality \(ab + bc + ca \leq a^2 + b^2 + c^2 \) can be rewritten as:
\[
a(b-a)+b(c-b)+c(a-c) \leq 0
\]

LIPS: LLM-based Inequality Prover with Symbolic Reasoning

- Summary: we develop an inequality proving system, where LLM and symbolic tools are used for rewriting and scaling the current inequality, respectively





Experimental Results

- Our system LIPS surpasses IMO Gold Medalists in inequality proving

	DeepSeek-R1	Gold medalists	LIPS
#Solved Olympiad-level Inequalities*	4/20	15/20	**16/20**

* Problems are collected from IMO competitions, national team selection test, training quizzes.

- LIPS achieves SoTA performance across various competition-level datasets

Dataset	# of Problems	Neural Provers			Symbolic Provers		LIPS	Δ
		DSP	MCTS	AIPS†	CAD‡	MMA‡		
ChenNEQ	41	0.0	17.0	-	70.7	68.2	**95.1**	24.4↑
MO-INT	20	0.0	15.0	50.0	60.0	60.0	**80.0**	20.0↑
567NEQ	100	0.0	4.0	-	54.0	52.0	**68.0**	14.0↑
Total	161	0.0	8.6	-	59.0	57.1	**76.3**	17.3↑

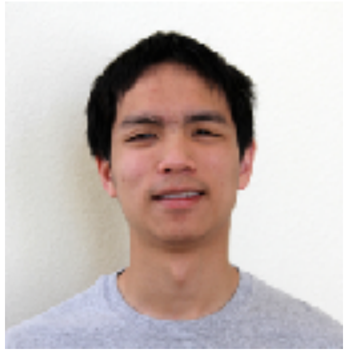
† The code of AIPS has not been publicly available, we only include its originally reported results.
‡ CAD and MMA only output verification results, they cannot produce human-readable proofs.

Some Interesting Findings

- LIPS finds novel proof paths expected to be impossible by human experts

Problem: Let a, b, c be three positive reals. Prove that if $abc = 1$, then

$$a^2 + b^2 + c^2 \geq a + b + c$$



Evan Chen
(IMO Coach for Team USA)

"AM-GM alone is hopeless here..."

Generated by LIPS

Formal solution:

```
theorem Example_1d7 (a b c : ℝ) (h : a * b * c = 1) : a + b + c ≤
    a ^ 2 + b ^ 2 + c ^ 2 := by
  scale NEQ_AM_GM_left_square_2vars (u := 1) (v := a) ...
  scale NEQ_AM_GM_left_square_2vars (u := 1) (v := c) ...
  scale NEQ_AM_GM_left_square_2vars (u := 1) (v := b) ...
  llm_rearrange ...
  llm_simplify ...  = 3/2 - a^2/2 - b^2/2 - c^2/2
  llm_rearrange (left := 3/2) (right := a^2/2 + b^2/2 + c^2/2)
  scale NEQ_AM_GM_right_normal_3vars (u := a^2/2) (v := b^2/2) ...
  llm_simplify ...  = (a*b*c)^2 / 8
  llm_simplify ...  = 1 / 8
  try close
```

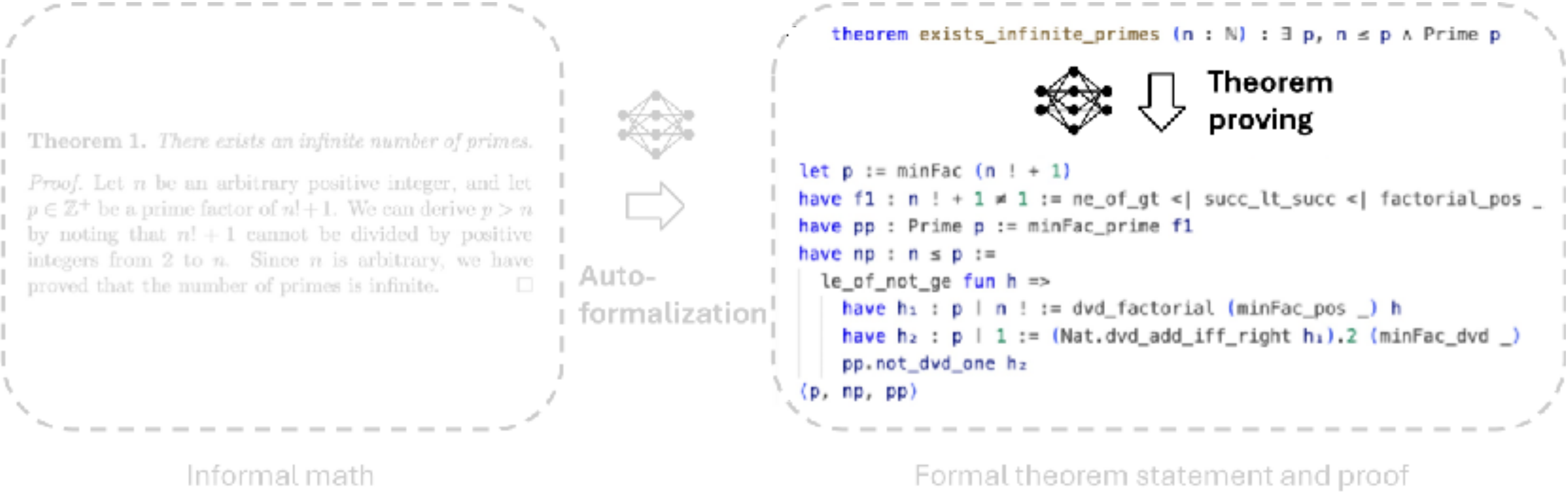
LIPS succeeds with exactly AM-GM

Source: https://web.evanchen.cc/handouts/Ineq/en.pdf

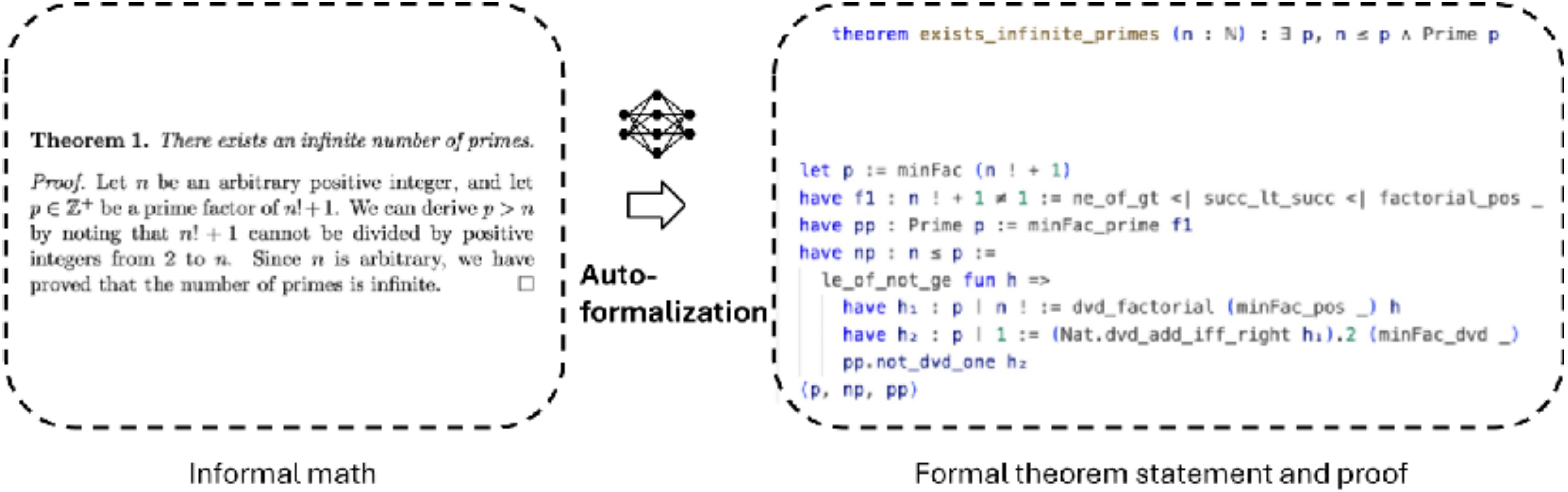
Takeaway

- Challenge in theorem proving: How to efficiently explore an infinite action space?
- Insights on a specific mathematical domain can be helpful
- Open problem: generalizing across different domains?

Theorem Proving



Autoformalization



[Wu et al. "Autoformalization with Large Language Models" NeurIPS 2022]

Autoformalizing Theorems and Proofs

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n! + 1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

Informal

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

Formal

Autoformalizing Theorems and Proofs

- Autoformalizing theorems: informal theorem → formal theorem

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. □

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

Informal

Formal

Autoformalizing Theorems and Proofs

- Autoformalizing theorems: informal theorem → formal theorem
- Autoformalizing proofs: informal theorem & proof + formal theorem → formal proof

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Hard to Evaluate Autoformalized Theorems

No reliable automatic evaluation

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Hard to Evaluate Autoformalized Theorems

No reliable automatic evaluation

Alternatives

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

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Hard to Evaluate Autoformalized Theorems

No reliable automatic evaluation

Alternatives

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

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

...

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Hard to Evaluate Autoformalized Theorems

No reliable automatic evaluation

- Equivalence checking is infeasible

Alternatives

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

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

...

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Hard to Evaluate Autoformalized Theorems

No reliable automatic evaluation

- Equivalence checking is infeasible
- Human evaluation is expensive
- Proxy metrics (e.g., BLEU) are inaccurate

Alternatives

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

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

...

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Reasoning Gaps in Informal Proofs

- Informal proofs have reasoning gaps
  - Explicit gaps: “left to the reader”
  - Implicit gaps
- Formal proofs must be gap-free

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. □

→

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

Informal

Formal

Key Challenges in Autoformalization

- Theorems: No reliable automatic evaluation
- Proofs: Reasoning gaps ubiquitous in informal proofs

Theorem 1. *There exists an infinite number of primes.*

Proof. Let n be an arbitrary positive integer, and let $p \in \mathbb{Z}^+$ be a prime factor of $n!+1$. We can derive $p > n$ by noting that $n!+1$ cannot be divided by positive integers from 2 to n. Since n is arbitrary, we have proved that the number of primes is infinite. $\square$

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

Informal

Formal

Key Challenges in Autoformalization

- Theorems: No reliable automatic evaluation
- Proofs: Reasoning gaps ubiquitous in informal proofs

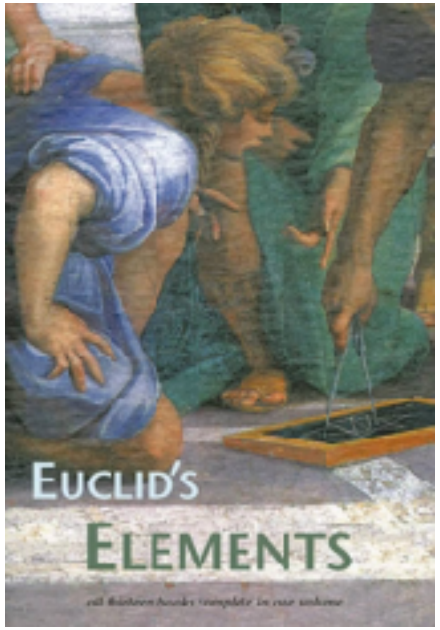
Things intractable in general can be made tractable in a specific domain

Informal

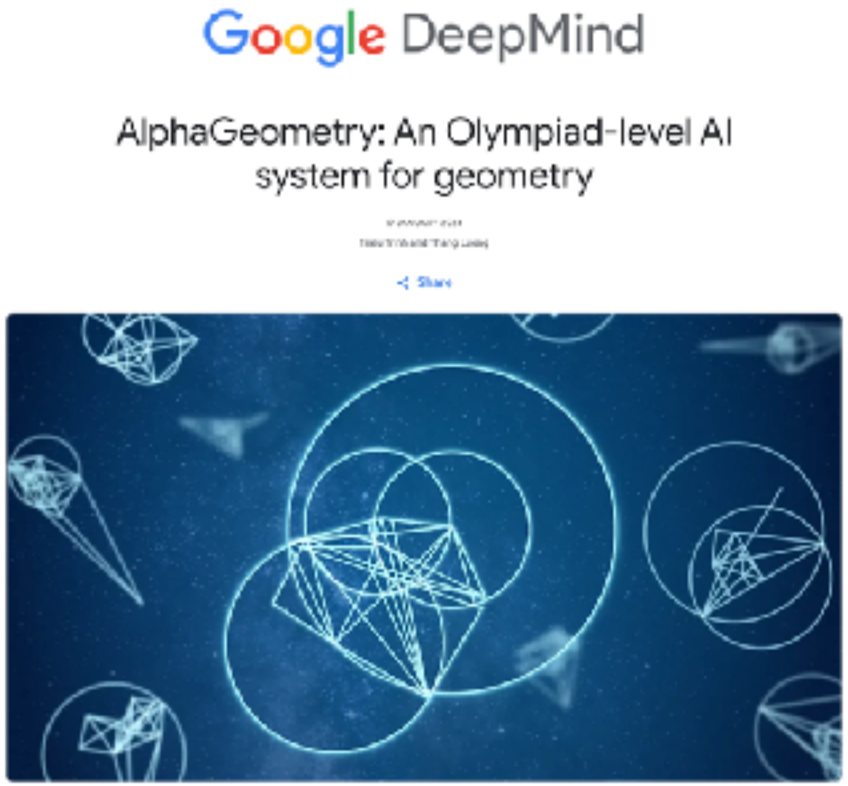
Formal

Euclidean Geometry

An arena for human and machine intelligence



Euclid (Εὐκλείδης), 300 BC



[Trinh *et al.*, **AlphaGeometry**, Nature 2024]

Autoformalizing Euclidean Geometry

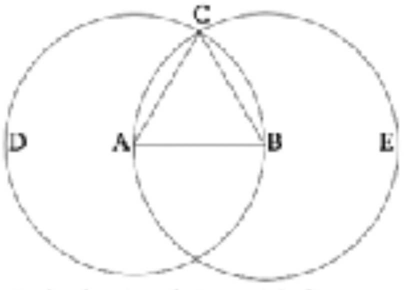
- LeanEuclid: Benchmark for autoformalizing Euclidean geometry
  - 48 from Euclid's Elements; 125 from UniGeo [Chen *et al.*, **UniGeo**, EMNLP 2022]

Autoformalizing Euclidean Geometry

- LeanEuclid: Benchmark for autoformalizing Euclidean geometry
  - 48 from Euclid's Elements; 125 from UniGeo   [Chen *et al.*, **UniGeo**, EMNLP 2022]

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,† to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

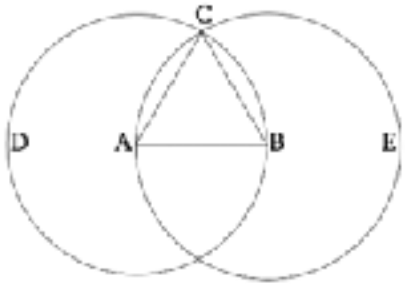
Informal theorem, proof, diagram

Autoformalizing Euclidean Geometry

- LeanEuclid: Benchmark for autoformalizing Euclidean geometry
    - 48 from Euclid's Elements; 125 from UniGeo [Chen *et al.*, **UniGeo**, EMNLP 2022]

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,[†] to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

→

```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
  ∃ c : Point, |(c─a)| = |(a─b)| ∧ |(c─b)| = |(a─b)|

by
  euclid_intros
  euclid_apply circle_from_points a b as BCD
  euclid_apply circle_from_points b a as ACE
  euclid_apply intersection_circles BCD ACE as c
  euclid_apply point_on_circle_onlyif a b c BCD
  euclid_apply point_on_circle_onlyif b a c ACE
  use c
  euclid_finish
```

Informal theorem, proof, diagram

Formal theorem & proof in Lean

Autoformalizing Euclidean Geometry

- LeanEuclid: Benchmark for autoformalizing Euclidean geometry
  - 48 from Euclid's Elements; 125 from UniGeo   [Chen *et al.*, **UniGeo**, EMNLP 2022]

Proposition 1

To construct an equilateral triangle on a given finite straight-line.

Let AB be the given finite straight-line.
So it is required to construct an equilateral triangle on the straight-line AB.
Let the circle BCD with center A and radius AB have

```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
```

First to faithfully formalize proofs in Euclid's Elements

constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

```
euclid_finish
```

Informal theorem, proof, diagram

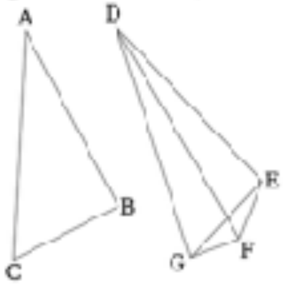
Formal theorem & proof in Lean

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24

Proposition 24

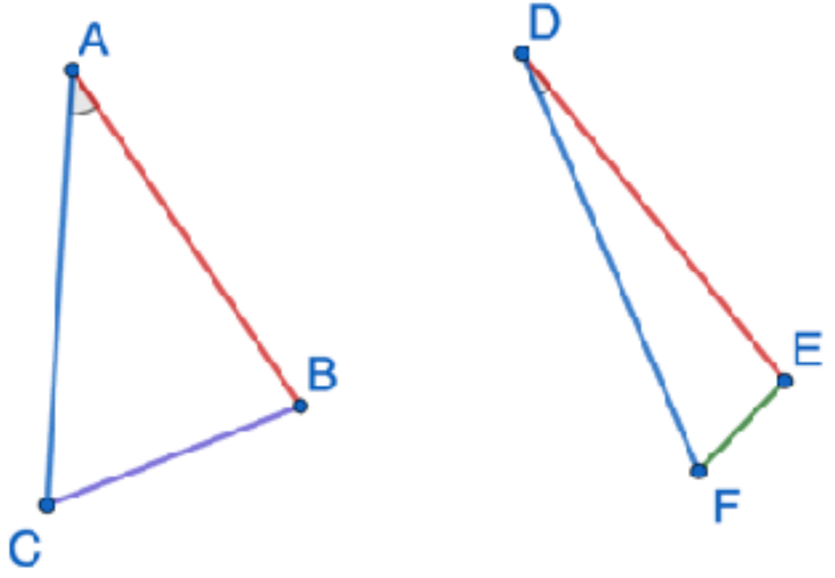
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

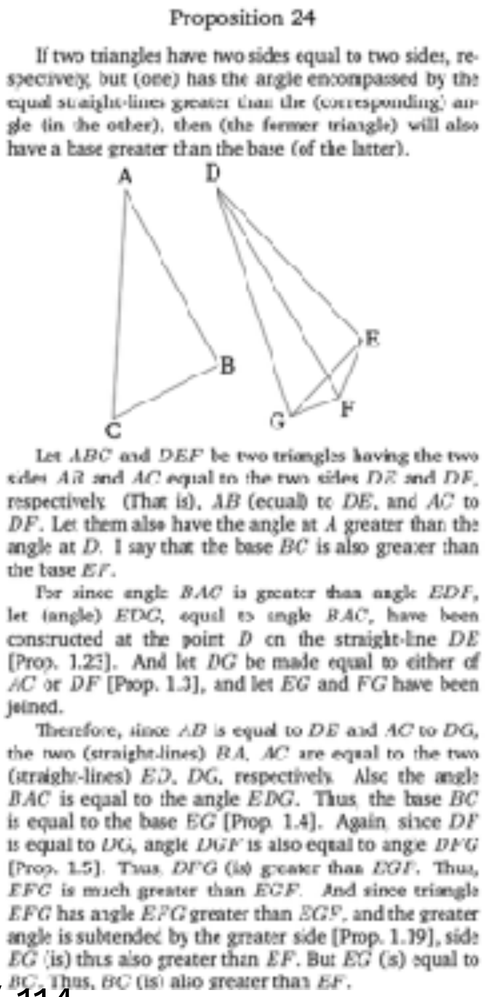
Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$$
\left.
\begin{aligned}
|AB| &= |DE| \\
|AC| &= |DF| \\
\angle BAC &> \angle EDF
\end{aligned}
\right.
\implies |BC| > |EF|
$$

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24



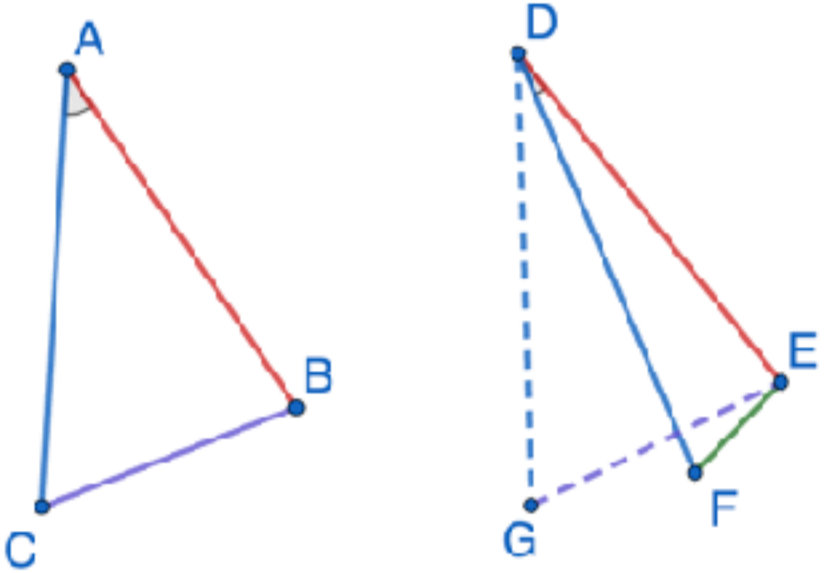
Proposition 24

If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).

Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



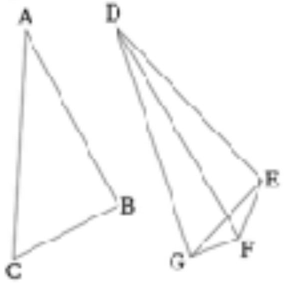
$$
\left.
\begin{array}{l}
|AB| = |DE| \\
|AC| = |DF| \\
\angle BAC > \angle EDF
\end{array}
\right.
\implies |BC| > |EF|
$$

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24

Proposition 24

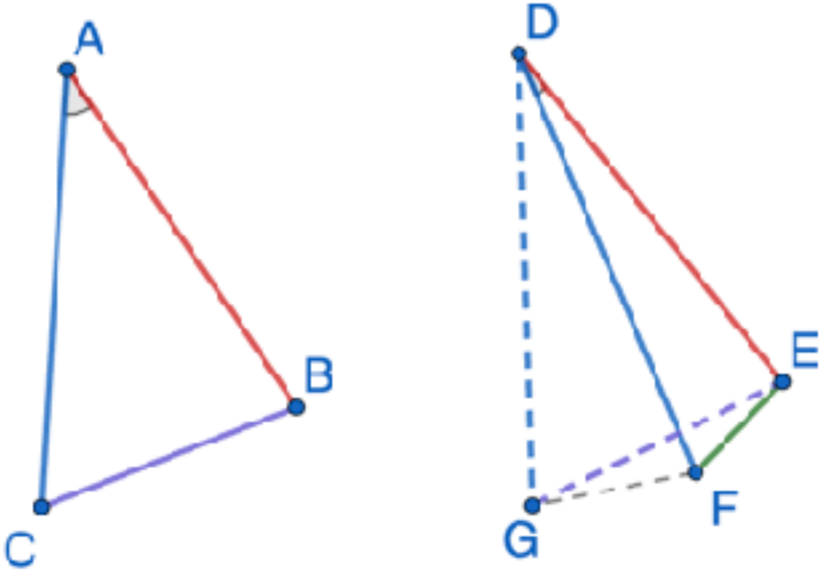
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$$
\begin{aligned}
|AB| &= |DE| \\
|AC| &= |DF| \\
\angle BAC &> \angle EDF
\end{aligned}
\quad \Longrightarrow \quad |BC| > |EF|
$$

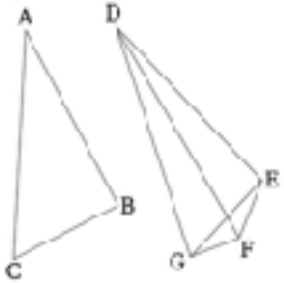
Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24

Only need to prove $\angle EFG > \angle EGF$

Proposition 24

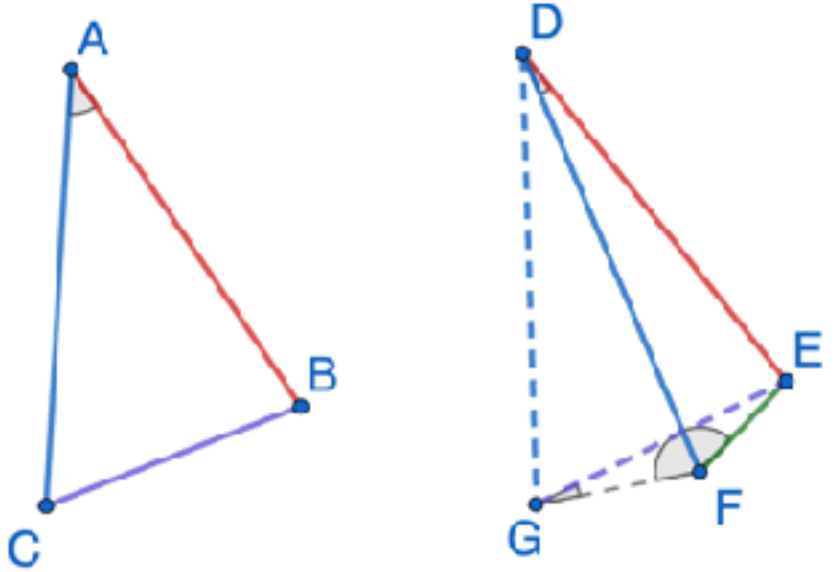
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



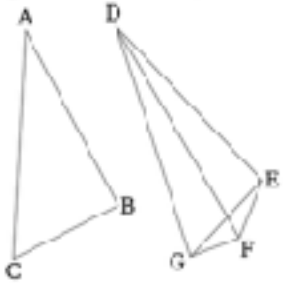
$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid's Proofs

Elements, Book I, Proposition 24

Proposition 24

If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

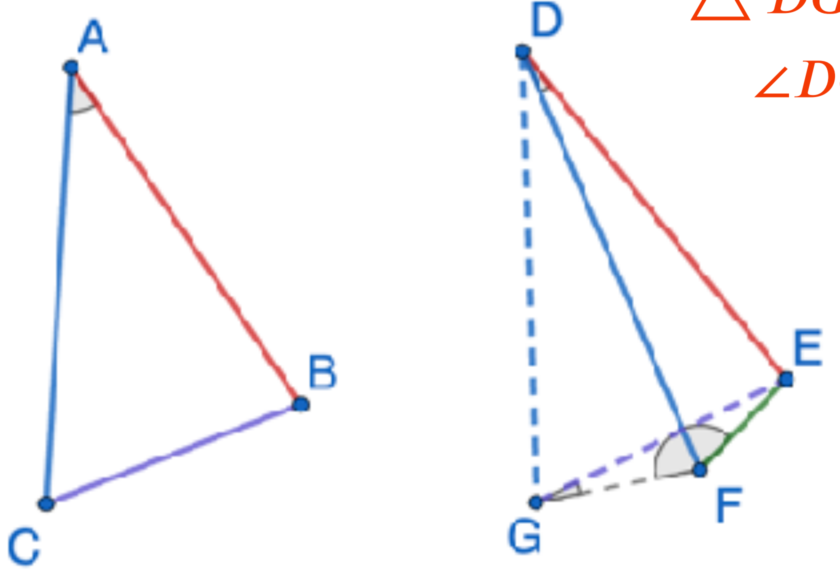
For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.

Only need to prove $\angle EFG > \angle EGF$

$\triangle\ DGF$ is isosceles!

$\angle DGF = \angle DFG$



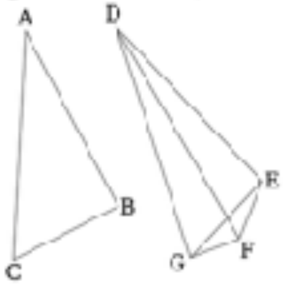
$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24

Proposition 24

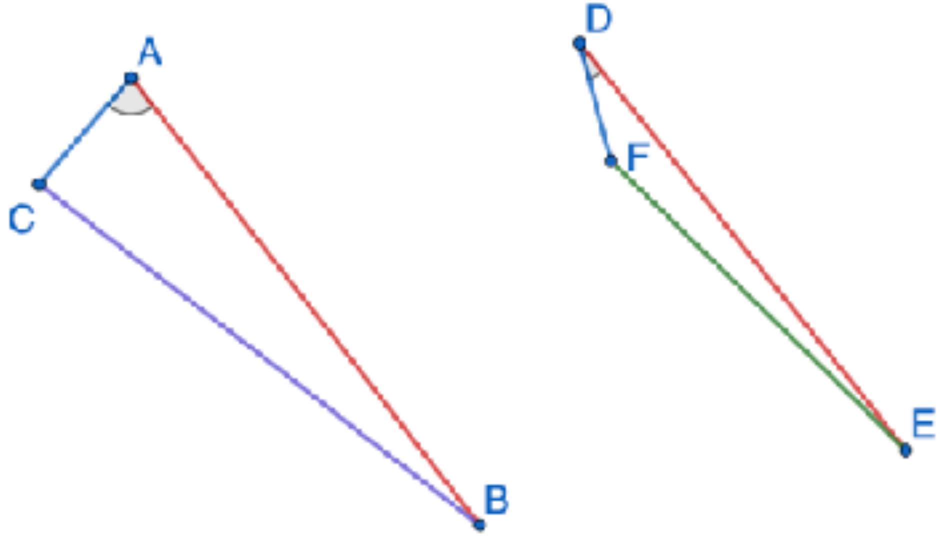
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



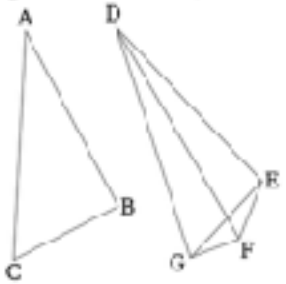
$$|AB| = |DE| \\ |AC| = |DF| \\ \angle BAC > \angle EDF \quad \Longrightarrow \quad |BC| > |EF|$$

Logical Gaps in Euclid's Proofs

Elements, Book I, Proposition 24

Proposition 24

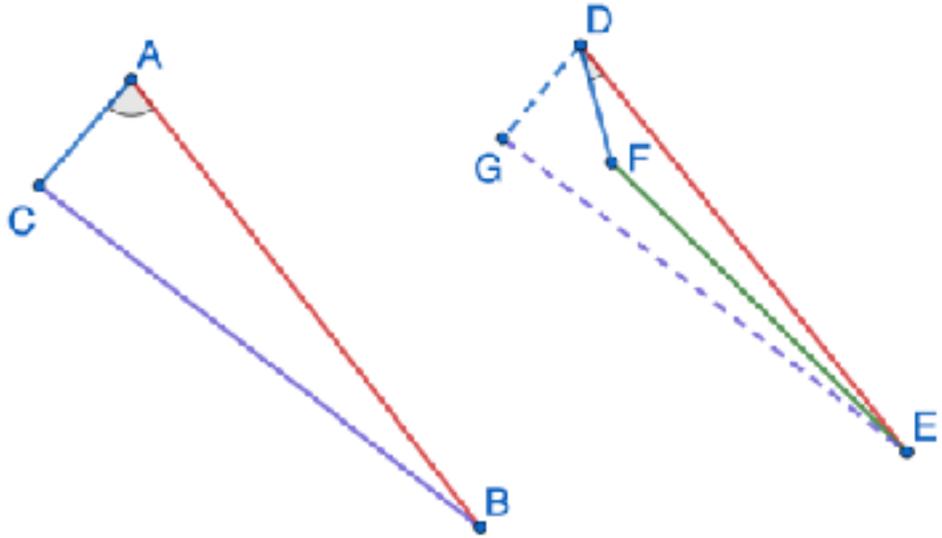
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



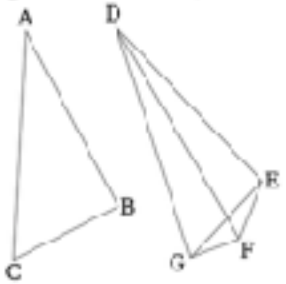
$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24

Proposition 24

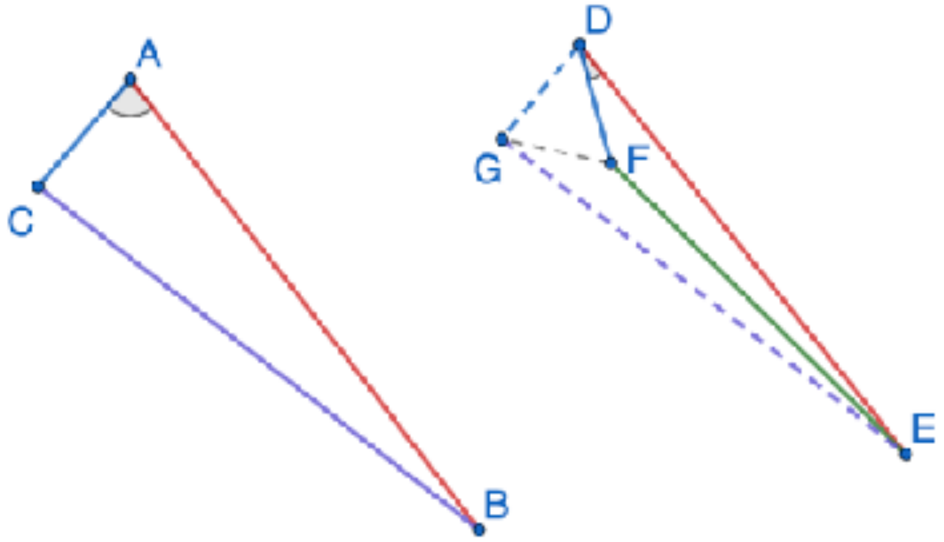
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid's Proofs

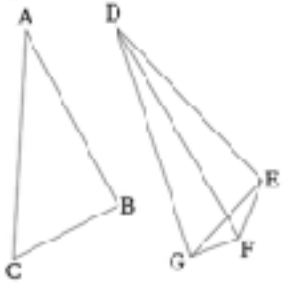
$\triangle\, DGF$ is isosceles

$\angle DGF = \angle DFG = \alpha$

Elements, Book I, Proposition 24

Proposition 24

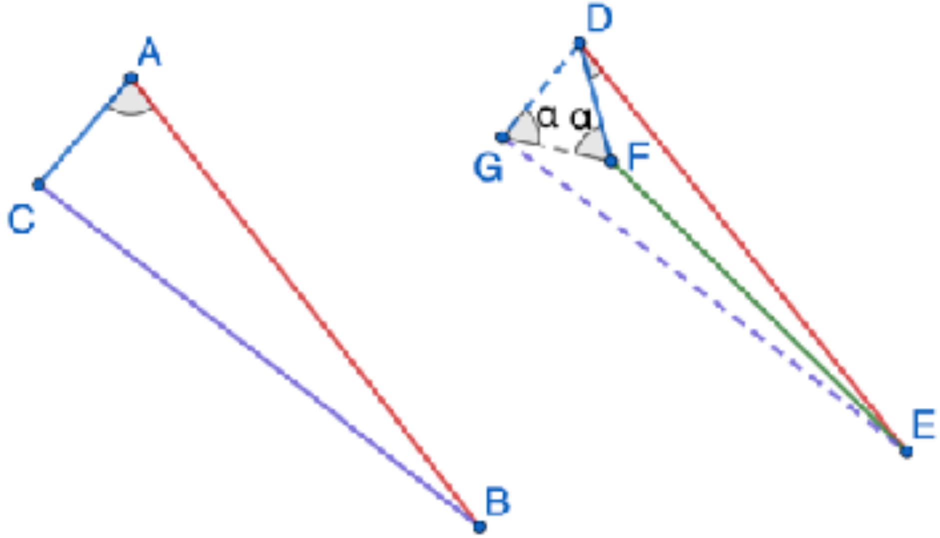
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid’s Proofs

$\triangle\ DGF$ is isosceles

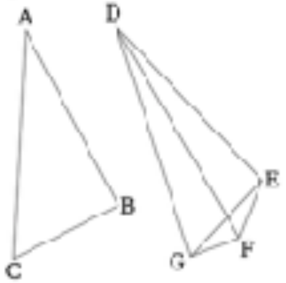
$\angle DGF = \angle DFG = \alpha$

Need to prove $x > y$

Elements, Book I, Proposition 24

Proposition 24

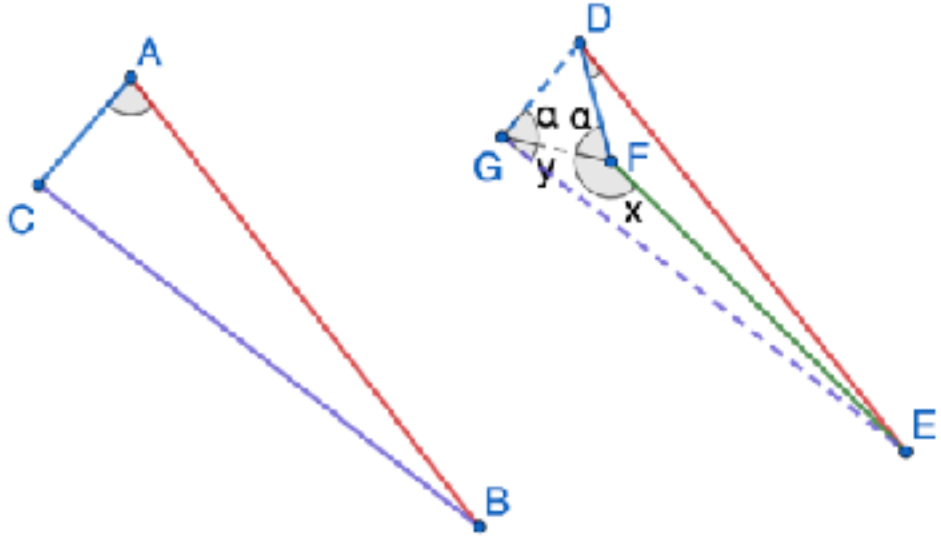
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



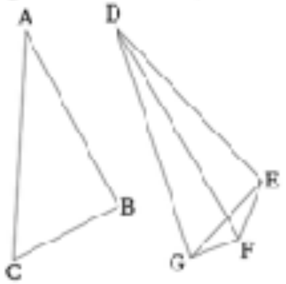
$$|AB| = |DE|$$
$$|AC| = |DF| \qquad \Longrightarrow \qquad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid's Proofs

Elements, Book I, Proposition 24

Proposition 24

If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

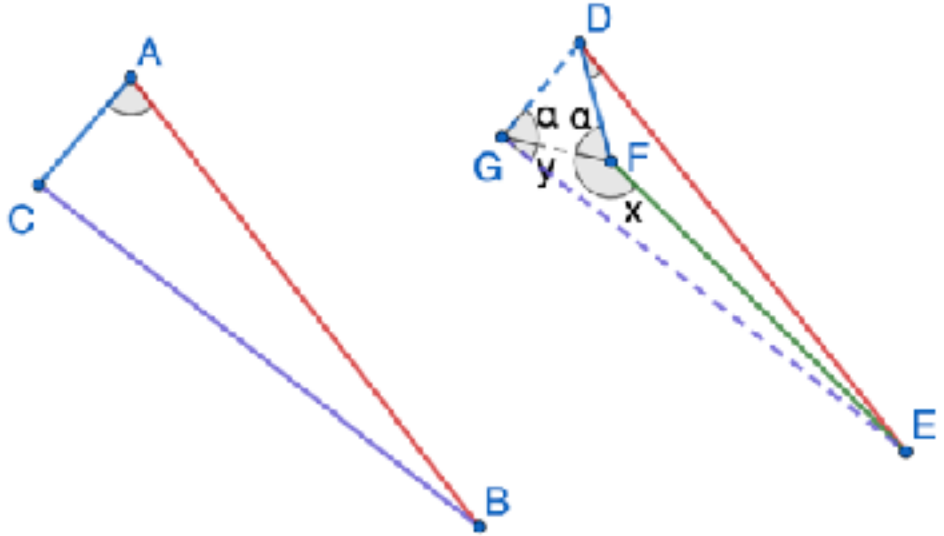
Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.

$\triangle\ DGF$ is isosceles

$\angle DGF = \angle DFG = \alpha$

Need to prove $x > y$

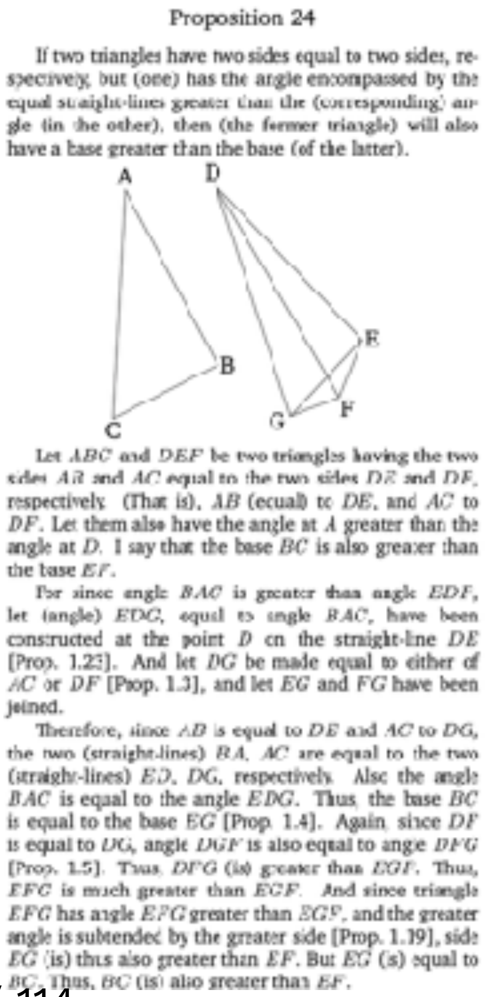
$\angle DGE = \alpha + y < \pi$



$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

Logical Gaps in Euclid’s Proofs

Elements, Book I, Proposition 24



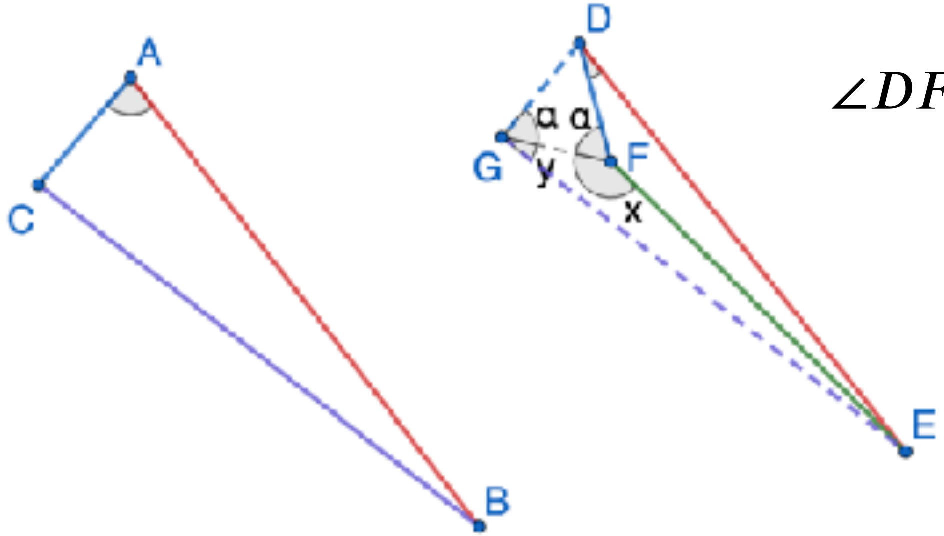
Proposition 24

If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).

Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$\triangle\, DGF$ is isosceles

$$\angle DGF = \angle DFG = \alpha$$

Need to prove $x > y$

$$\angle DGE = \alpha + y < \pi$$

$$\angle DFE = 2\pi - \alpha - x < \pi$$

$$|AB| = |DE|$$
$$|AC| = |DF|$$
$$\angle BAC > \angle EDF$$

$$\Longrightarrow$$

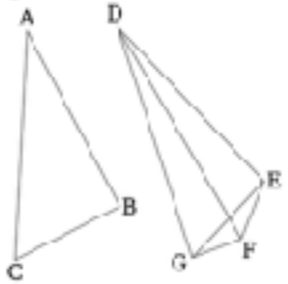
$$|BC| > |EF|$$

Logical Gaps in Euclid's Proofs

Elements, Book I, Proposition 24

Proposition 24

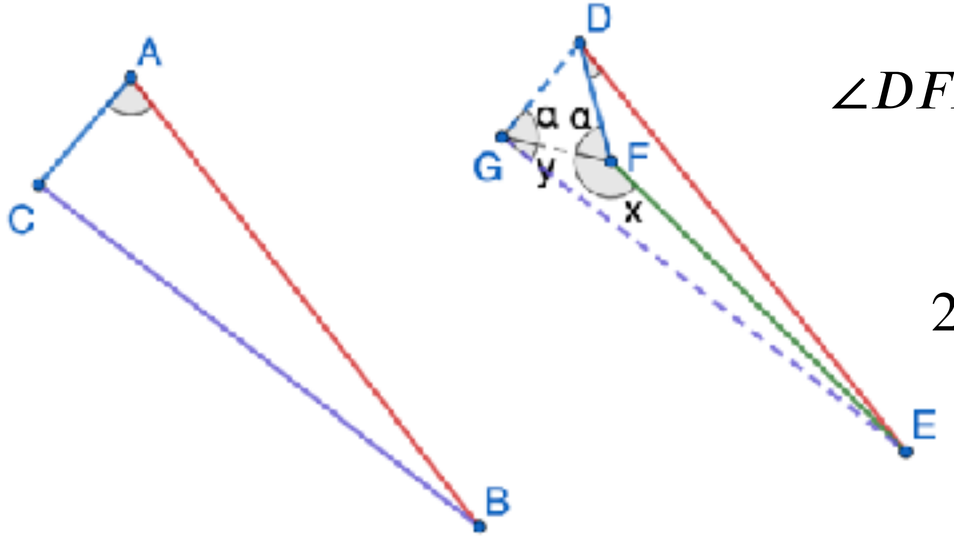
If two triangles have two sides equal to two sides, respectively, but (one) has the angle encompassed by the equal straight-lines greater than the (corresponding) angle (in the other), then (the former triangle) will also have a base greater than the base (of the latter).



Let ABC and DEF be two triangles having the two sides AB and AC equal to the two sides DE and DF, respectively. (That is), AB (equal) to DE, and AC to DF. Let them also have the angle at A greater than the angle at D. I say that the base BC is also greater than the base EF.

For since angle BAC is greater than angle EDF, let (angle) EDG, equal to angle BAC, have been constructed at the point D on the straight-line DE [Prop. 1.23]. And let DG be made equal to either of AC or DF [Prop. 1.3], and let EG and FG have been joined.

Therefore, since AB is equal to DE and AC to DG, the two (straight-lines) BA, AC are equal to the two (straight-lines) ED, DG, respectively. Also the angle BAC is equal to the angle EDG. Thus, the base BC is equal to the base EG [Prop. 1.4]. Again, since DF is equal to DG, angle DGF is also equal to angle DFG [Prop. 1.5]. Thus, DFG (is) greater than EGF. Thus, EFG is much greater than EGF. And since triangle EFG has angle EFG greater than EGF, and the greater angle is subtended by the greater side [Prop. 1.19], side EG (is) thus also greater than EF. But EG (is) equal to BC. Thus, BC (is) also greater than EF.



$$|AB| = |DE|$$
$$|AC| = |DF| \quad \Longrightarrow \quad |BC| > |EF|$$
$$\angle BAC > \angle EDF$$

$\triangle\ DGF$ is isosceles

$$\angle DGF = \angle DFG = \alpha$$

Need to prove $x > y$

$$\angle DGE = \alpha + y < \pi$$

$$\angle DFE = 2\pi - \alpha - x < \pi$$

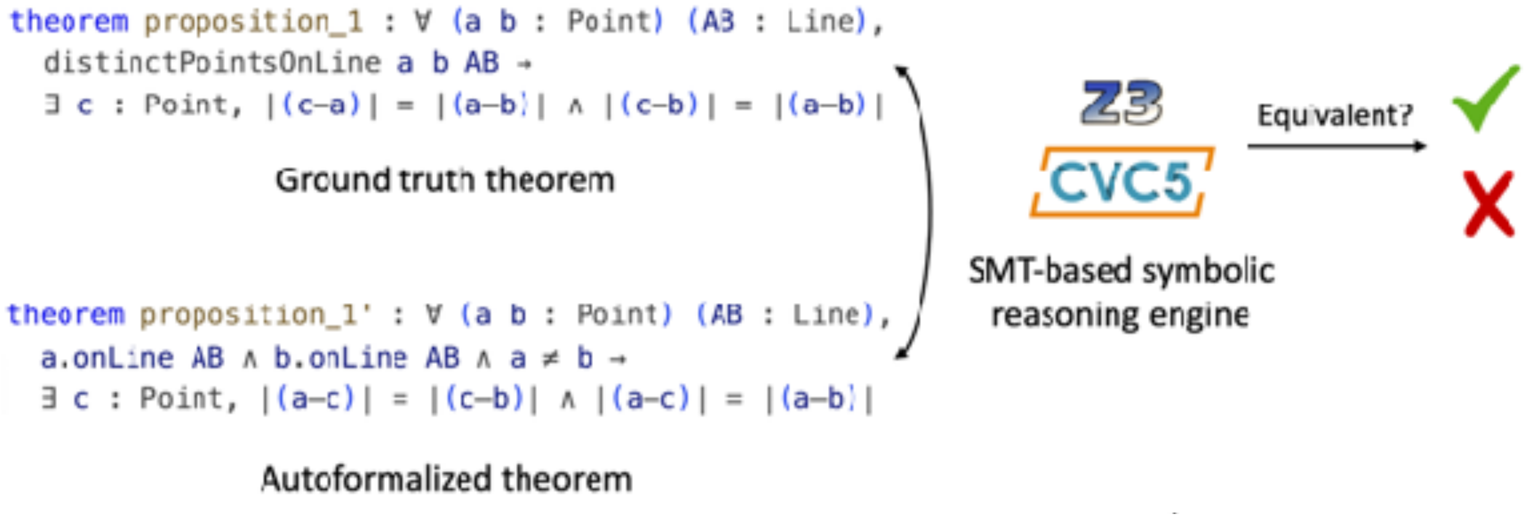
$$\Downarrow$$

$$2\pi - x + y < 2\pi$$

Q.E.D.

Equivalence Checking Between Theorems

- Two theorems T_1 and T_2 are equivalent iff we can prove $T_1 \iff T_2$
- Symbolic reasoning engine based on SMT solvers



Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid’s Elements, Book I, Proposition 1

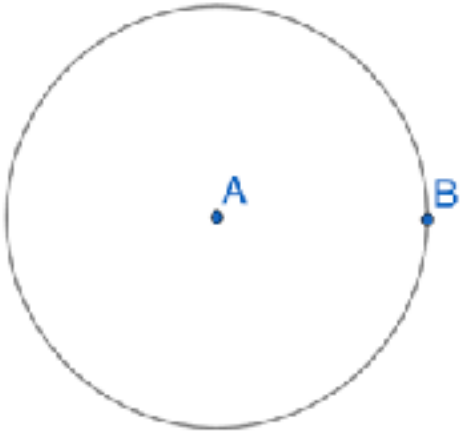
One can construct a equilateral triangle given two distinct points

Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid's Elements, Book I, Proposition 1

One can construct a equilateral triangle given two distinct points

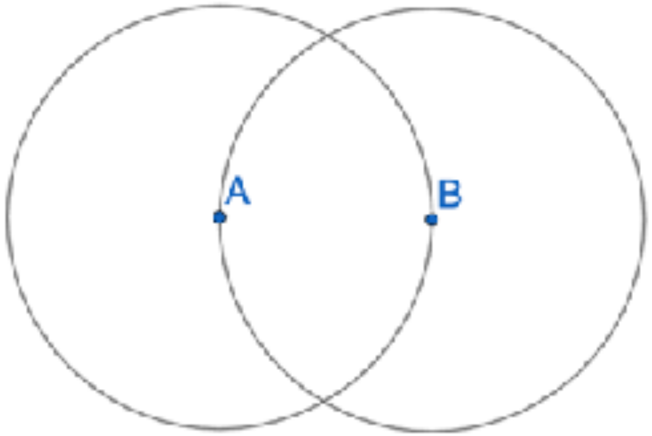


Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid’s Elements, Book I, Proposition 1

One can construct a equilateral triangle given two distinct points

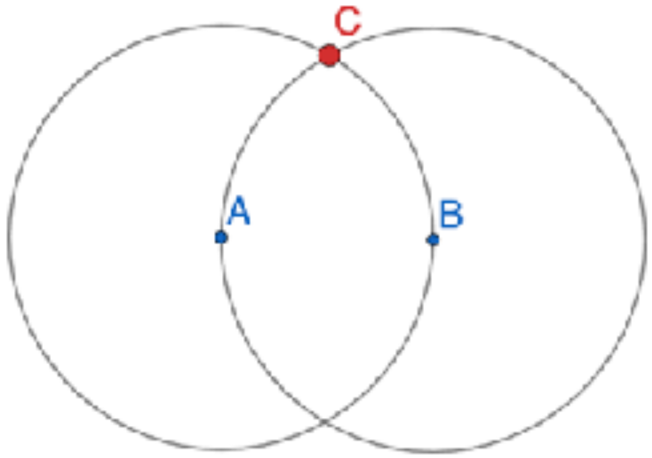


Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid’s Elements, Book I, Proposition 1

One can construct a equilateral triangle given two distinct points

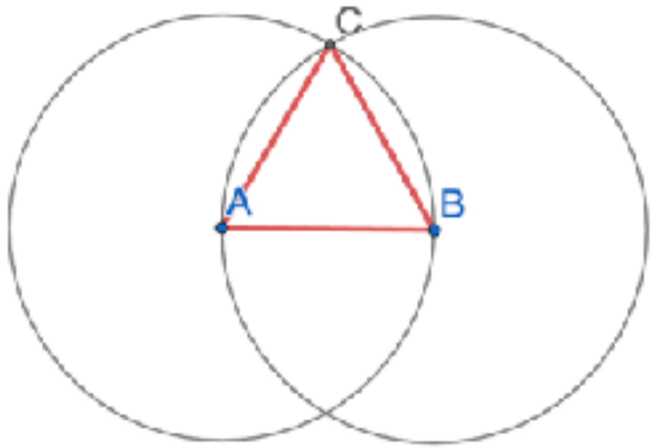


Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid's Elements, Book I, Proposition 1

One can construct a equilateral triangle given two distinct points

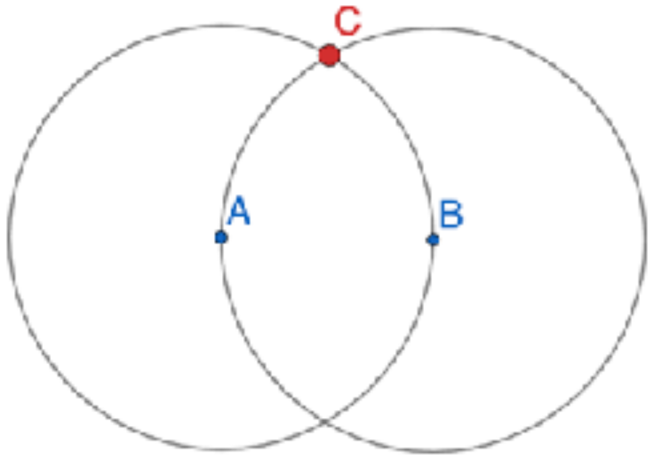


Diagrammatic Reasoning

Ubiquitous reasoning gaps in Euclidean geometry

- Geometry proofs rely on diagrams that are hard to formalize
- Example: Euclid's Elements, Book I, Proposition 1

One can construct a equilateral triangle given two distinct points



Did we prove C exists?

Modeling Diagrammatic Reasoning

The Formal System E

[Avigad *et al.*, "A formal system for Euclid's Elements", 2008]

- Diagrammatic reasoning are logical consequences of "diagrammatic rules”

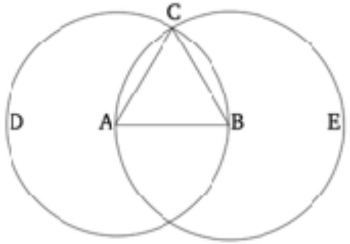
```
centre unique : ∀ (a b : Point) (α : Circle), (isCentre c α) ∧ (isCentre b α) → a = b

center_inside_circle : ∀ (a : Point) (α : Circle), isCentre c α → insideCircle a α

inside_not_on_circle : ∀ (a : Point) (α : Circle), insideCircle a α → ¬(onCircle a α)

between_symm : ∀ (a b c : Point), between a b c → (between c b a) ∧ (a ≠ b) ∧ (a ≠ c) ∧
    ¬(between b a c)

between_same_line_out : ∀ (a b c : Point) (L : Line), (between a b c) ∧ (onLine a L) ∧ (
    onLine b L) → onLine c L

between same line in : ∀ (a b c : Point) (L : Line), (between a b c) ∧ (onLine a L) ∧ (
    onLine c L) → onLine b L
```

- We imple SMT solvers

Putting It Together

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,[†] to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

Informal Euclidean geometry problem

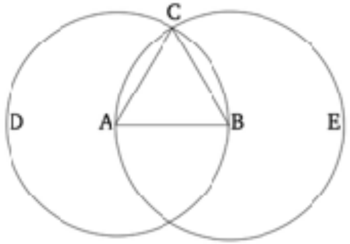
```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
  ∃ c : Point, |(c─a)| = |(a─b)| ∧ |(c─b)| = |(a─b)|
```

Ground truth theorem

Putting It Together

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,[†] to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

Informal Euclidean geometry problem

⇨

```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
  ∃ c : Point, |(c─a)| = |(a─b)| ∧ |(c─b)| = |(a─b)|
```

Ground truth theorem

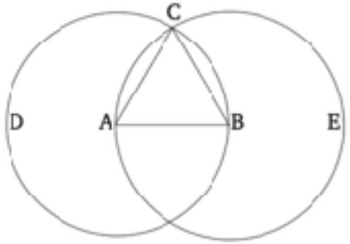
```
theorem proposition_1' : ∀ (a b : Point) (AB : Line),
  a.onLine AB ∧ b.onLine AB ∧ a ≠ b →
  ∃ c : Point, |(a─c)| = |(c─b)| ∧ |(a─c)| = |(a─b)|
```

Autoformalized theorem

Putting It Together

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,[†] to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

Informal Euclidean geometry problem

```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
  ∃ c : Point, |(c─a)| = |(a─b)| ∧ |(c─b)| = |(a─b)|
```

Ground truth theorem

```
theorem proposition_1' : ∀ (a b : Point) (AB : Line),
  a.onLine AB ∧ b.onLine AB ∧ a ≠ b →
  ∃ c : Point, |(a─c)| = |(c─b)| ∧ |(a─c)| = |(a─b)|
```

Autoformalized theorem

Z3

CVC5

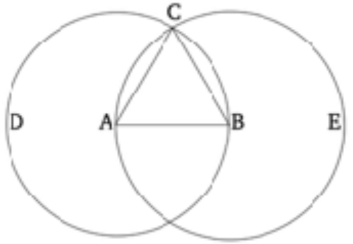
SMT-based symbolic reasoning engine

Equivalent?

Putting It Together

Proposition 1

To construct an equilateral triangle on a given finite straight-line.



Let AB be the given finite straight-line.

So it is required to construct an equilateral triangle on the straight-line AB.

Let the circle BCD with center A and radius AB have been drawn [Post. 3], and again let the circle ACE with center B and radius BA have been drawn [Post. 3]. And let the straight-lines CA and CB have been joined from the point C, where the circles cut one another,[†] to the points A and B (respectively) [Post. 1].

And since the point A is the center of the circle CDB, AC is equal to AB [Def. 1.15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 1.15]. But CA was also shown (to be) equal to AB. Thus, CA and CB are each equal to AB. But things equal to the same thing are also equal to one another [C.N. 1]. Thus, CA is also equal to CB. Thus, the three (straight-lines) CA, AB, and BC are equal to one another.

Thus, the triangle ABC is equilateral, and has been constructed on the given finite straight-line AB. (Which is) the very thing it was required to do.

Informal Euclidean geometry problem

```
theorem proposition_1 : ∀ (a b : Point) (AB : Line),
  distinctPointsOnLine a b AB →
  ∃ c : Point, |(c─a)| = |(a─b)| ∧ |(c─b)| = |(a─b)|
```

Ground truth theorem

```
theorem proposition_1' : ∀ (a b : Point) (AB : Line),
  a.onLine AB ∧ b.onLine AB ∧ a ≠ b →
  ∃ c : Point, |(a─c)| = |(c─b)| ∧ |(a─c)| = |(a─b)|
```

Autoformalized theorem

Z3

CVC5

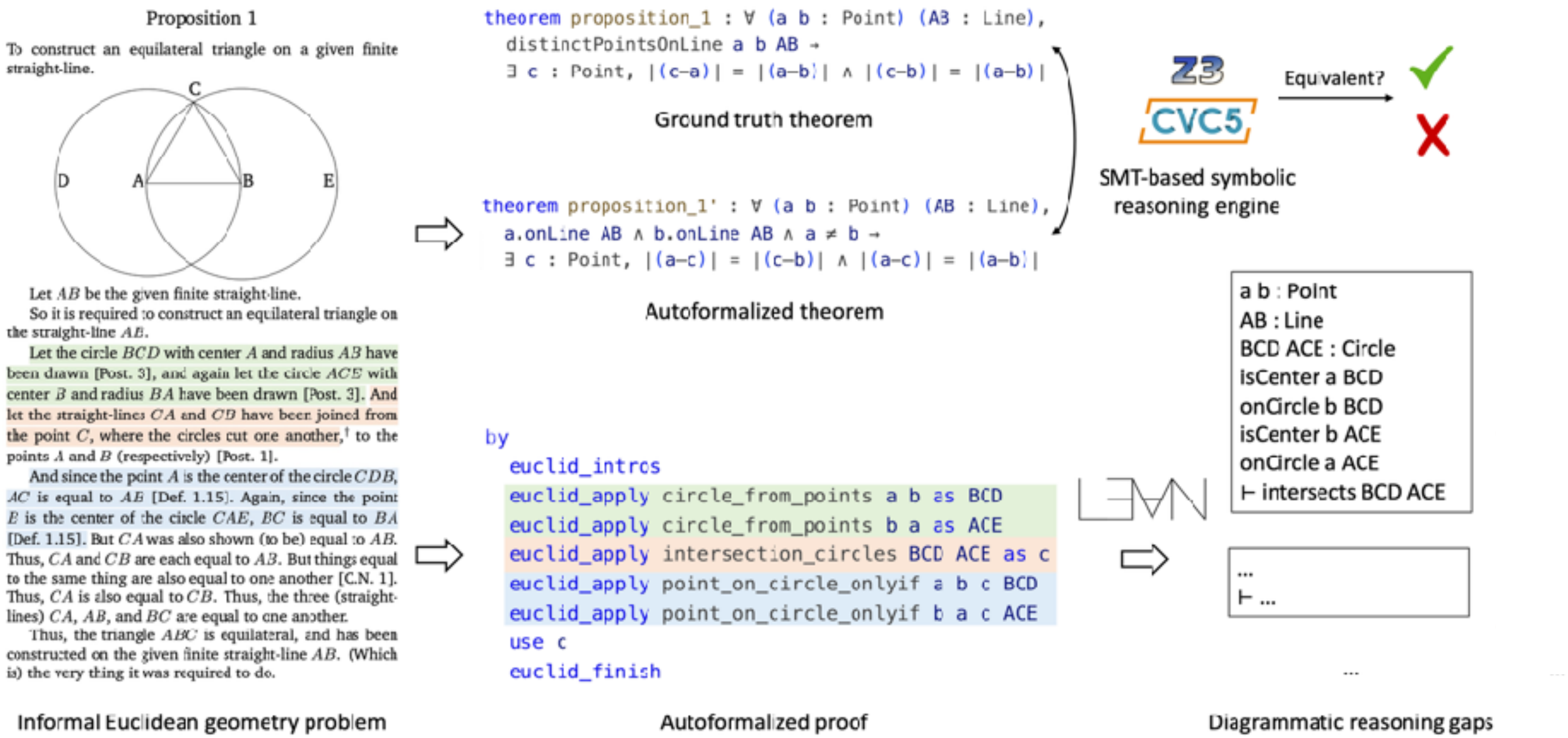
Equivalent?

SMT-based symbolic reasoning engine

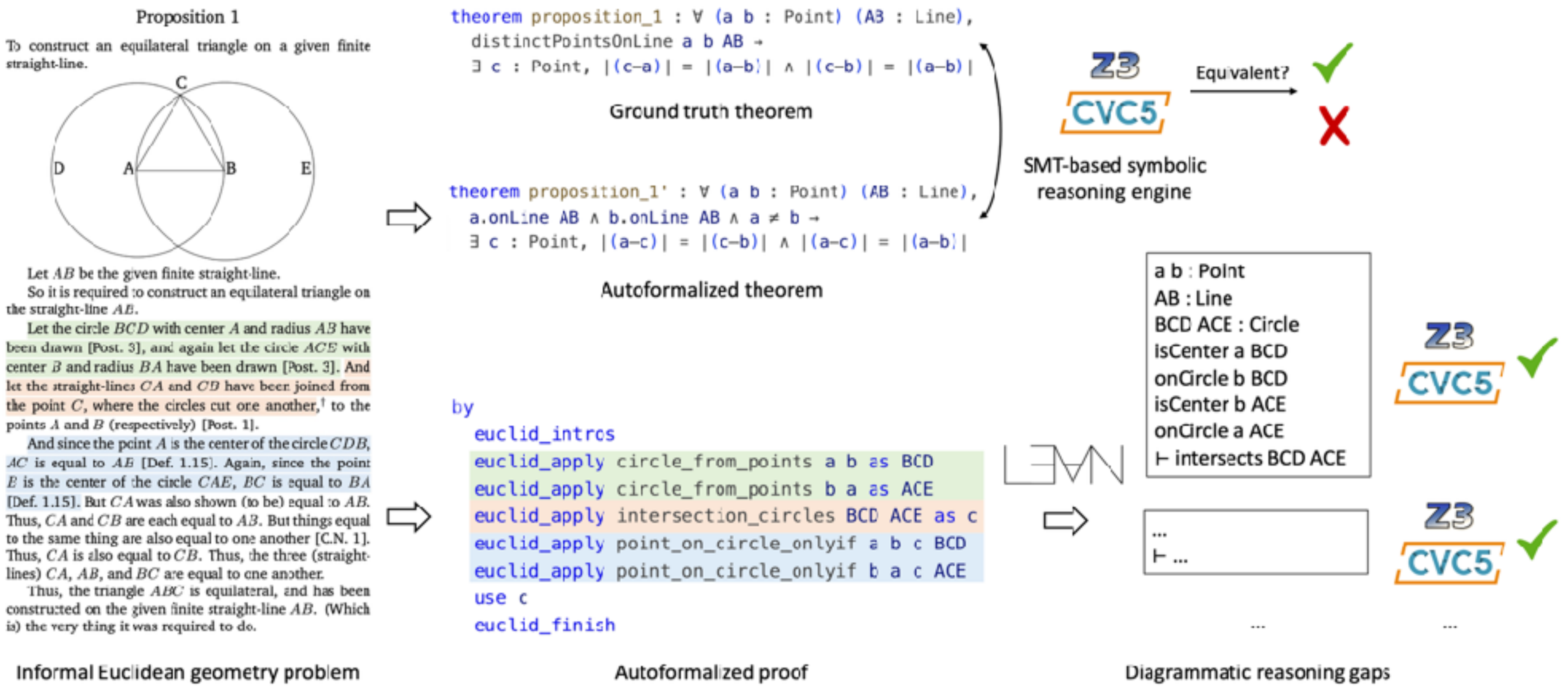
```
by
  euclid_intros
  euclid_apply circle_from_points a b as BCD
  euclid_apply circle_from_points b a as ACE
  euclid_apply intersection_circles BCD ACE as c
  euclid_apply point_on_circle_onlyif a b c BCD
  euclid_apply point_on_circle_onlyif b a c ACE
  use c
  euclid_finish
```

Autoformalized proof

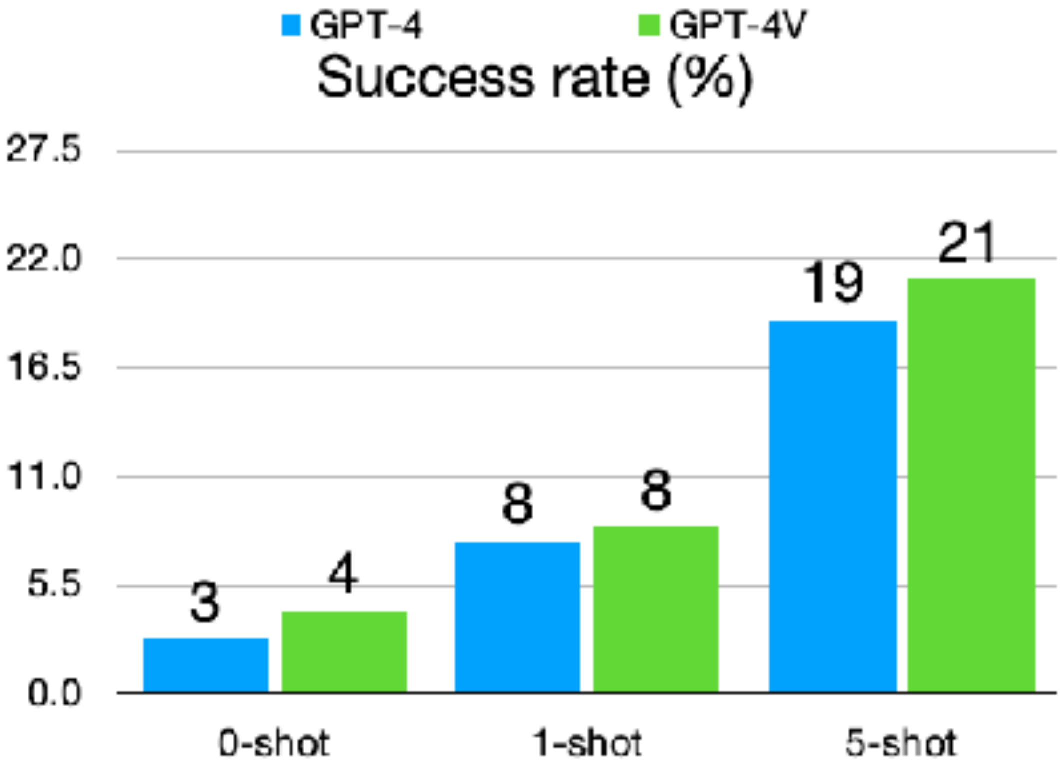
Putting It Together



Putting It Together



Experiments: Autoformalizing Theorems



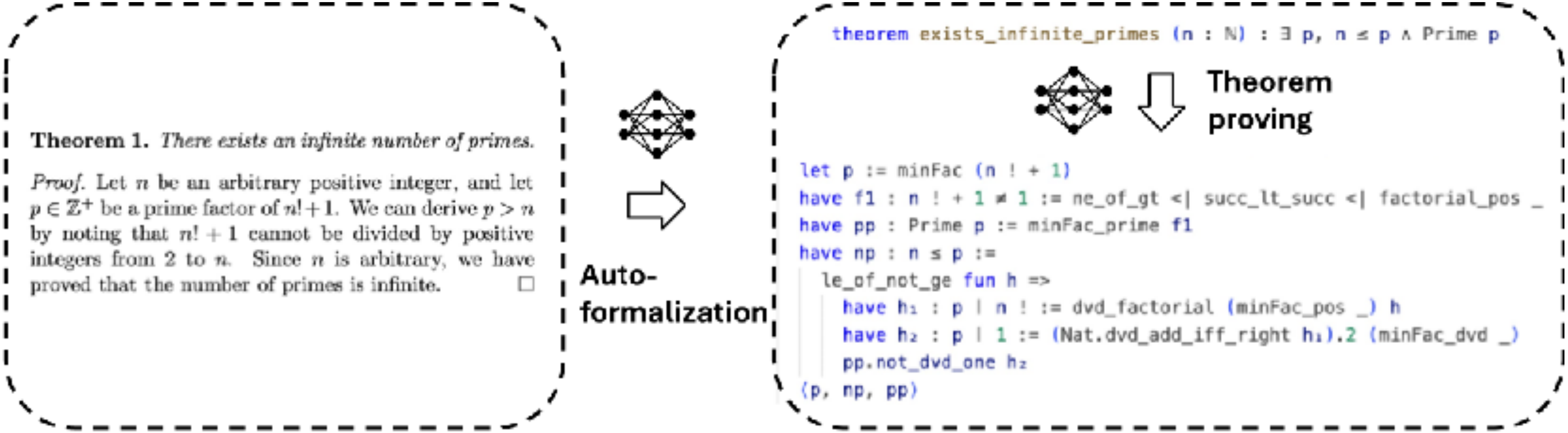
Takeaways

- Two challenges in autoformalization
  - Autoformalized theorems are difficult to evaluate
  - Autoformalizing proofs require filling in reasoning gaps

- They can be addresses leveraging knowledge in specific domains

- Open problem: How to generalize across domains?

AI Meets Formal Mathematics



Informal math

Formal theorem statement and proof