Bridging Informal and Formal Mathematical Reasoning

Sean Welleck

May 28, 2025

Carnegie Mellon University

AI in expert domains

AI in expert domains

- Finance
- Science
- Mathematics

AI in expert domains

AI in expert domains

- Finance
- Science
- Mathematics
  - Open-ended dialogue
  - Come up with counterexamples
  - Help write proofs
  - ...

“Informal” mathematics

Math as raw data (text, images, ...)

- Flexible
- Widely used
- Difficult to check

Input:
Let r be a root of $x^2 + 5x + 7 = 0$. Compute $(r-1)(r+2)(r+6)(r+3)$.

Llemma 34B solution:
We have

$$\begin{aligned}(r-1)(r+2)(r+6)(r+3) &= (r^2 + r - 2)(r^2 + 9r + 18)\\ &= r^4 + 10r^3 + 25r^2 + 19r - 36\\ &= r^4 + 5r^3 + 7r^2 + 5r + 7 + 5r^3 + 20r^2 + 14r - 36\\ &= r^4 + 5r^3 + 25r^2 + 19r - 29\\ &= (r^2 + 5r + 7)(r^2 + 4r - 5)\\ &= (0)(r^2 + 4r - 5)\\ &= \boxed{0}.\end{aligned}$$

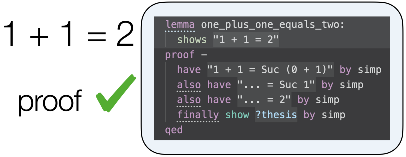
Final Answer: The final answer is 0.

Language model solution.

Formal mathematics

Math as **source code**

- Write a specification (e.g., 1+1=2)
- Write a proof
- **Automatically checked**
  - Code compiles ≡ correct proof



Math as source code.

Formal mathematics

Math as **source code**

- Write a specification (e.g., $1+1=2$)
- Write a proof
- **Automatically checked**
  - Code compiles $\equiv$ correct proof

Lean Isabelle Coq

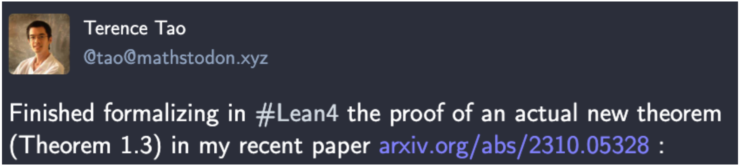
Theorem proving languages

Formal mathematics (Demo)

If $R \subseteq S$ and $S \subseteq T$ then $R \subseteq T$

Formal math for mathematics

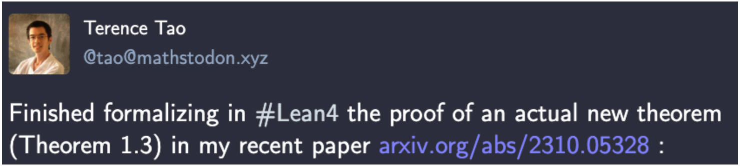
Growing use in mathematics:



Terence Tao's Lean formalization project (October 2023)

Formal math for mathematics

Growing use in mathematics:



Terence Tao's Lean formalization project (October 2023)

- New form of **collaboration**: break a big problem into multiple pieces

Formal math for AI/LLMs

Formal math for AI

- Tests **reasoning**
  - From easy: $1+1 = 2$
  - To hard: Fermat's Last Theorem

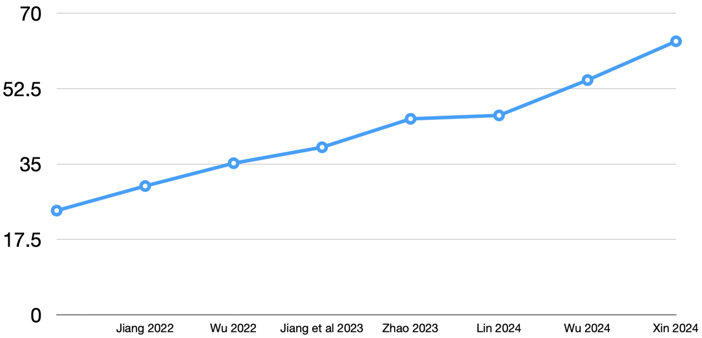
Formal math for AI/LLMs

Formal math for AI

- Tests **reasoning**
  - From easy: $1+1 = 2$
  - To hard: Fermat's Last Theorem
- **Verifiable**
  - Feedback signal for reinforcement learning

LLMs ∩ formal math

Rapid progress in methods based on language models:



miniF2F benchmark performance, 2022-2024

LLMs ∩ formal math

```
theorem imo_1960_p2 (x : ℝ) (h₀ : 0 ≤ 1 + 2 * x) (h₁ : (1 - Real.sqrt (1 + 2 *
    x)) ^ 2 ≠ 0)
    (h₂ : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 < 2 * x + 9) : -(1 / 2)
   ≤ x ∧ x < 45 / 8 := by
  norm_num at h₀ h₁ h₂
  have h₃ : 0 ≤ 1 + 2 * x := by linarith
  have h₄ : 0 < 1 + Real.sqrt (1 + 2 * x) := by
    nlinarith [Real.sqrt_nonneg (1 + 2 * x)]
  have h₅ : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 < 2 * x + 9 := by
   linarith
  have h₆ : 1 - Real.sqrt (1 + 2 * x) ≠ 0 := by
    intro h
    apply h₁
    nlinarith
  have h₇ : 4 * x ^ 2 / (1 - Real.sqrt (1 + 2 * x)) ^ 2 = (1 + Real.sqrt (1 +
   2 * x)) ^ 2 := by
    field_simp [h₆]
    nlinarith [sq_sqrt (show 0 ≤ 1 + 2 * x by linarith)]
  rw [h₇] at h₅
  constructor <;> nlinarith [sq_sqrt (show 0 ≤ 1 + 2 * x by linarith)]
```

Generated International Math Olympiad solution in Lean
(DeepSeek Prover-1.5B, Xin et al 2024)

Terence Tao
@tao@mathstodon.xyz

Finished formalizing in #Lean4 the proof of an actual new theorem (Theorem 1.3) in my recent paper arxiv.org/abs/2310.05328 :

The ability of Github copilot to correctly anticipate multiple lines of code for various routine verifications, and inferring the direction I want to go in from clues such as the names I am giving the theorems, continues to be uncanny.

Terence Tao's Lean formalization project (October 2023)

LLMs $\cap$ formal math

Terence Tao
@tao@mathstodon.xyz

Finished formalizing in #Lean4 the proof of an actual new theorem (Theorem 1.3) in my recent paper arxiv.org/abs/2310.05328 :

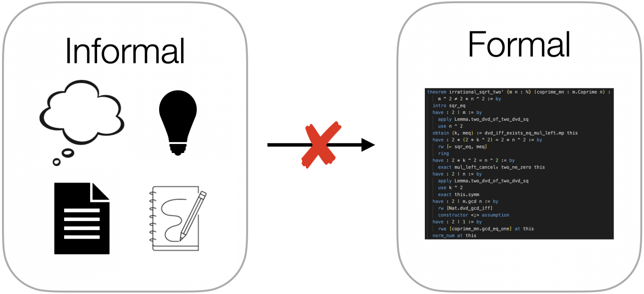
The ability of Github copilot to correctly anticipate multiple lines of code for various routine verifications, and inferring the direction I want to go in from clues such as the names I am giving the theorems, continues to be uncanny.

Terence Tao's Lean formalization project (October 2023)

So...why don't people and LLMs always use formal math?

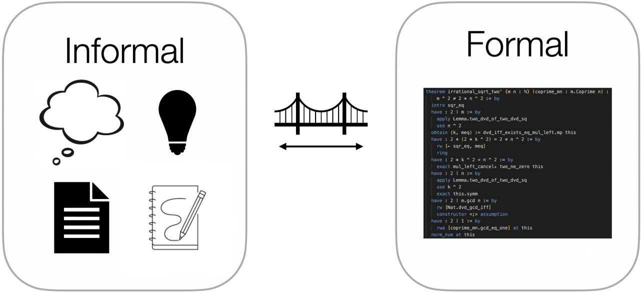
Key challenge: the informal-formal gap

Informal ideas, intuitions, and even proofs are difficult to express formally:



- Each step of reasoning needs to be specified in detail
- Requires a deep knowledge of the formal system

Bridging Informal and Formal Mathematical Reasoning



Flexible

Easy to check

This talk: Bridging Informal and Formal Mathematical Reasoning

This talk: Bridging Informal and Formal Mathematical Reasoning

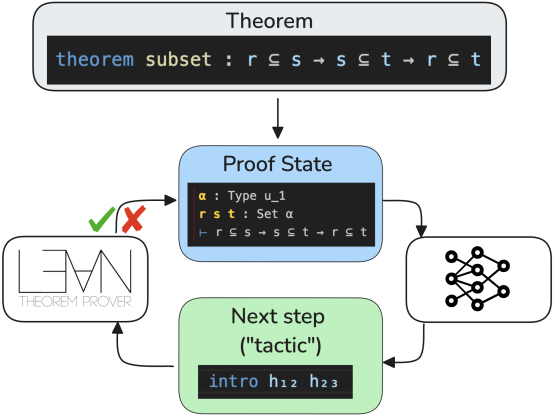
1. Informal thoughts
2. Informal sketches
3. Towards research-level mathematics

I: Informal thoughts

1. Training models to "think" — Lean-STaR

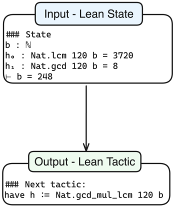
Lean-STaR: Learning to Interleave Thinking and Proving
Haohan Lin, Zhiqing Sun, Yiming Yang, Sean Welleck
ICLR 2025 (**Spotlight**)

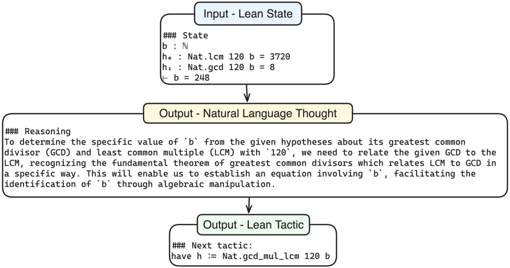
Neural theorem proving



- Math as checkable code
- Proof: sequence of (state, step)

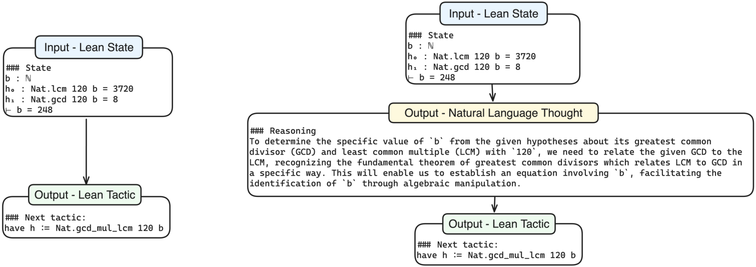
1. Training models to “think” — Lean-STaR





Can we train a model to “think” before each step of formal reasoning?

1. Training models to “think” — Lean-STaR



Why?

- Plan proof steps
- Diversify search space
- More tokens can give more computational capacity

1. Training models to "think" — Lean-STaR

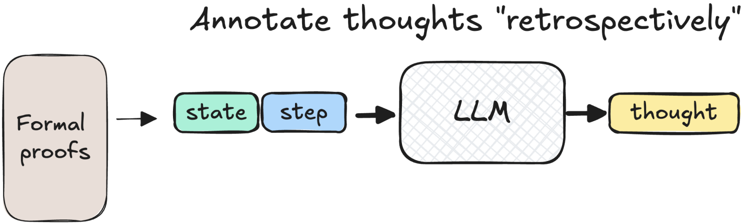
Lean-STaR (Self-taught reasoner[1])

Learn to generate thoughts via reinforcement learning

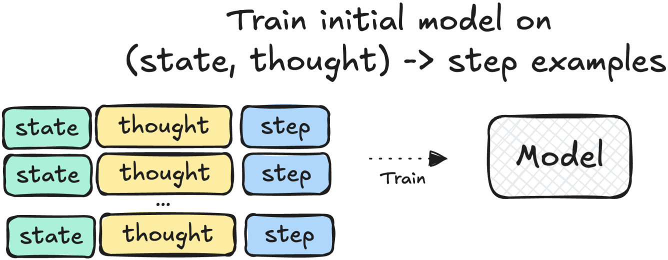
1. Initialization
2. Reinforcement learning

[1]Inspired by *STaR: Bootstrapping Reasoning with Reasoning*, Zelikman et al 2022

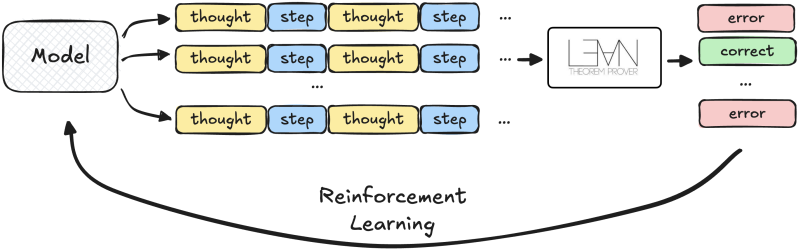
1. **Initialization**



1. **Initialization**



2: **Reinforcement learning**



Training models to "think" — Lean-STaR

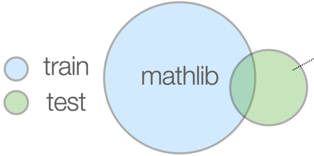
Algorithm: train on the successful proofs, and repeat:[2]

- Collect (state, thought, tactic) from successful proofs
- Train a new model $p_\theta^1(\mathit{thought}, \mathit{tactic}|\mathit{state})$
- Generate proofs
- ...

[2] I.e. Expert Iteration [Polu et al 2022], Rest-EM [Singh et al 2024]

Training models to “think” — Lean-STaR

- **miniF2F**: competition problems (AMC, AIME, IMO)

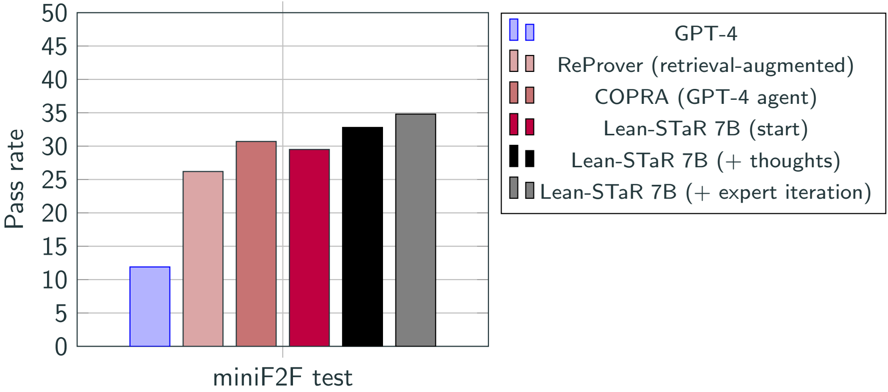


Problem **1959 IMO Problems/Problem 1**

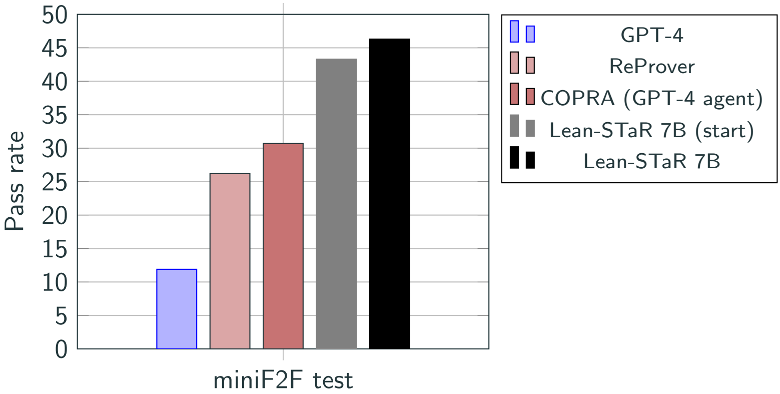
Prove that the fraction $\frac{21n+4}{14n+3}$ is irreducible for every natural number n.

```
theorem imo_1959_p1
  (n : ℕ)
  (h₀ : 0 < n) :
  Nat.gcd (21*n + 4) (14*n + 3) = 1 := by sorry
```

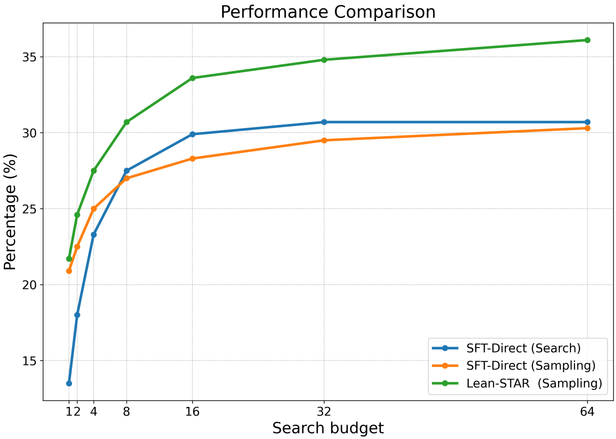
Training models to “think” — Lean-STaR



Training models to "think" — Lean-STaR



Training models to “think” — Lean-STaR



Increasing the search budget is more effective with thoughts

Training models to “think” — Lean-STaR

```
theorem amc12a_2002_p6 (n : ℕ) (h₀ : 0 < n)
    : ∃ m, m > n ∧ ∃ p, m * p ≤ m + p :=
    use n + 1, by linarith only [h₀] -- Reasoning: To demonstrate the existence of
     such an `m` and `p` that satisfy the given inequalities, we leverage the
     property of natural numbers that ensures an increase in `n` from `0` leads to
     the possibility of finding such `m` and `p`. This approach is supported by the
     fundamental property of natural numbers and their arithmetic operations,
     suggesting that for a given `n` greater than `0`, there exists a successor `m`
     and a number `p` that satisfy the conditions outlined.
    exact ⟨0, by simp ⟩ -- Reasoning: To demonstrate that there exists an \(n \in ℕ
     \) such that \((n + 1) * p \leq n + 1 + p\) holds for some \(p \in ℕ\), we
     recognize that setting \(p = 0\) satisfies the inequality by the properties of
     zero in arithmetic. This is because \(n + 1 > 0\) by assumption, ensuring the
     left-hand side is non-negative and the right-hand side simplifies to \(n +
     1\), making the inequality trivially true.
```

Example generated thoughts and proof from Lean-STaR

Training models to “think” — Lean-STaR

```
theorem aime_1983_p2 (x p : ℝ) (f : ℝ → ℝ) (h₀ : 0 < p ∧ p < 15) (h₁ : p ≤ x ∧ x
    ≤ 15)
    (h₂ : f x = abs (x - p) + abs (x - 15) + abs (x - p - 15))
    : 15 ≤ f x :=
    cases' h₁ with h₁ h₂ -- Reasoning: Given a function \(f\) defined over the real
    numbers, where \(f(x)\) equals the sum of the absolute values of the
    differences \(x - p\), \(x - 15\), and \(x - p - 15\), along with conditions
    on \(p\) and \(x\) that ensure \(x\) lies between \(p\) and \(15\) and \(p\)
    is positive, we aim to show that \(f(x) \geq 15\). To demonstrate this, we
    consider all possible cases for the position of \(x\) relative to \(p\) and
    \(15\), specifically at \(p\) itself, between \(p\) and \(15\), and at \(15\).
    By evaluating \(f(x)\) for each case, we can establish a lower bound of
    \(f(x)\) that is greater than or equal to \(15\).
    simp [h₀, abs_of_nonneg (sub_nonneg.2 h₁), abs_of_nonneg (sub_nonneg.2 h₂), *]
    -- Reasoning: The objective is to show that \(f(x)\) is at least 15 for \(x\)
    within the bounds \(p \le x \le 15\), where \(p\) is positive but less than
    15. The provided expression for \(f(x)\) suggests using properties of absolute
    values and inequalities to simplify and analyze \(f(x)\). By applying
    simplification rules for absolute values and inequalities, along with the
    given bounds, we aim to demonstrate the desired inequality \(f(x) \ge 15\).
    simp only [abs_of_nonneg] at * -- Reasoning: The objective is to establish an
    inequality involving an expression with absolute values and a sum, where all
    terms are functions of \(x\). By recognizing that the absolute value functions
    are always non-negative, we can simplify the given expression by removing the
    absolute value symbols, thus focusing on the arithmetic manipulation of the
    terms to demonstrate the inequality.
    simp only [abs_of_nonneg, sub_add_eq_add_sub] at * -- Reasoning: The objective
    is to demonstrate that the expression `15 ≤ x - p + |x - 15| + |x - p - 15|`
    holds given the conditions on `p`, `x`, and the function `f`. Since the
    expression in question involves absolute values which inherently involve
    non-negativity, manipulating these expressions to highlight their
    non-negativity or equality is a logical initial step. This involves
    recognizing that the absolute value of a difference can be rewritten as the
    difference of absolute values, which simplifies the manipulation of the
    expression to prove the inequality.
    congr 1 -- Reasoning: To establish the inequality `15 ≤ x + |x - 15| + |x - p
    - 15| - p`, we recognize that the left-hand side of the inequality involves a
    mix of additions and subtractions of real numbers. By introducing a congruence
    transformation that simplifies operations involving absolute values, we aim to
    align both sides of the inequality in a form that directly reflects the
    properties of real numbers under these operations.
    simp only [abs_of_nonpos (sub_nonpos_of_le h₂), abs_of_nonneg (sub_nonneg_of_le
    h₁)] -- Reasoning: The objective is to prove an inequality involving absolute
    values and subtractions that aligns the given conditions and constraints.
    Simplification using the properties of absolute values and the provided bounds
    can directly lead to the desired conclusion by expressing the absolute values
    in terms of their underlying expressions and applying the given constraints.
    simp [abs_of_pos, *] -- Reasoning: To prove the given inequality involving
    absolute values, simplification using algebraic properties and given
    hypotheses will directly lead us to the desired inequality by recognizing the
    algebraic manipulation and application of the known bounds `h₁` and `h₂`, as
    well as the non-negativity of `p`.
    rw [abs_of_nonpos] -- Reasoning: The objective is to demonstrate that `15 ≤ 15
    + |x - p - 15| - p` under the given constraints on `p` and `x`. Given `p < 15`
    and `p ≤ x ≤ 15`, it's apparent that `p - 15` is non-positive. This
    observation allows us to apply the property that the absolute value of a
    non-positive number is equal to its negative value, thereby simplifying the
    expression on the right-hand side of the inequality.
    all_goals linarith -- Reasoning: The goal involves demonstrating a relationship
    between expressions involving inequalities and subtraction. The simplification
    of these expressions into a form that directly compares their numerical values
    can lead to a straightforward application of known inequalities and algebraic
    properties, demonstrating the inequality's validity under the given conditions.
```

Example generated thoughts and proof from Lean-STaR

Training models to "think" — After Lean-STaR

After Lean-STaR, incorporating thoughts became a widely-used component of LLM-based theorem proving:

deepseek

DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search

Huajian Xin*, Z.Z. Ren*, Junxiao Song*, Zhihong Shao*, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang
Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z.F. Wu, Fuli Luo, Chong Ruan

DeepSeek-AI

Output:

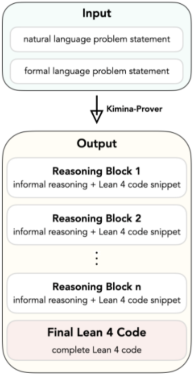
```
  -- This is equivalent to showing that 9x^2 sin^2 x - 12x sin x + 4 ≥ 0, and the
    left-hand side can be rewritten as a perfect square (3x sin x - 2)^2.
  -- We use the fact that (3x sin x - 2)^2 is non-negative to establish this.
  nlinarith [sq_nonneg (3 * x * Real.sin x - 2)]
`````
```

generating theorem proof code. Similar to Lean-STaR (Lin et al., 2024), which performs isolated chain-of-thought reasoning (Wei et al., 2022; Feng et al., 2023) before each proof step, our method integrates this reasoning directly as comments within the proof code. We use the DeepSeek-

Informal thoughts in DeepSeek Prover 1.5
```

Training models to "think" — After Lean-STaR

After Lean-STaR, incorporating thoughts became a widely-used component of LLM-based theorem proving:



KIMINA-PROVER PREVIEW: TOWARDS LARGE FORMAL REASONING MODELS WITH REINFORCEMENT LEARNING

TECHNICAL REPORT OF KIMINA-PROVER PREVIEW

Numina & Kimi Team

Next, we'll substitute $d = \frac{15}{2}$ back into one of the equations to solve for a. Let's use h_0: $a + 6d = 30$

$a + 6(\frac{15}{2}) = 30$

$a + 45 = 30$

$a = 30 - 45 = -15$

In Lean, once we have d, we can determine a:

```
have ha : a = -15 := by
  linarith [h₀, hd]
```

Now we need to compute $a + 20d$: $a + 20d = -15 + 20(\frac{15}{2}) = -15 + 150 = 135$

In Lean, we can directly compute $a + 20d$ using the values of a and d:

```
linarith [ha, hd]
```

Informal thoughts in Kimina Prover (April 14 2025)

Training models to "think" — After Lean-STaR

More broadly, *reasoning models* that generate long chains-of-thought have subsequently begun to revolutionize LLM reasoning:

September 12, 2024 Release

Learning to reason with LLMs

We are introducing OpenAI o1, a new large language model trained with reinforcement learning to perform complex reasoning. o1 thinks before it answers—it can produce a long internal chain of thought before responding to the user.

OpenAI o1 reasoning model

deepseek

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

DeepSeek R1 reasoning model

Training models to “think” — Lean-STaR

Recap: **Lean-STaR**

- Training on formal code may be insufficient to learn the underlying thought process needed to produce the code
- Learn to generate thoughts via reinforcement learning

This talk: Bridging Informal and Formal

1. Informal thoughts
2. **Informal sketches**
3. Towards research-level mathematics

II: Informal sketches

Motivation: informal proofs and formal proofs

Statement

Prove that n is 70 if $\gcd(n, 40) = 10$ and $\operatorname{lcm}(n, 40) = 280$.

Informal proof

We know that $\gcd(a, b) \cdot \operatorname{lcm}(a, b) = ab$, hence $10 \cdot 280 = n \cdot 40$.

Then $n = 10 \cdot 280/40 = 70$,

completing the proof. ■

How would we write this as a formal proof?

Motivation: informal proofs and formal proofs

Informal proof

We know that $\gcd(a,b) \cdot \mathrm{lcm}(a,b) = ab$, hence $10 \cdot 280 = n \cdot 40$.

Then $n = 10 \cdot 280/40 = 70$,

completing the proof. ■

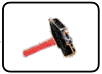
Formal proof

```
have c1: "10*280 = n*40"
using assms
    by (smt (z3) prod_gcd_lcm_nat)
then have c2: "n = 10*280/40"
    by auto
then show ?thesis
    by auto
```

A proof with a high-level sketch and low-level proof steps.

Low-level provers: Sledgehammer

```
have c1: “10*280 = n*40”
using assms
```

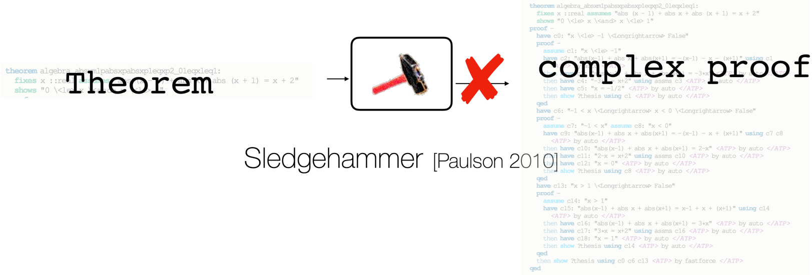


```
have c1: “10*280 = n*40”
using assms
by (smt (z3) prod_gcd_lcm_nat)
```

Sledgehammer [Paulson 2010] calls out to external automated provers.

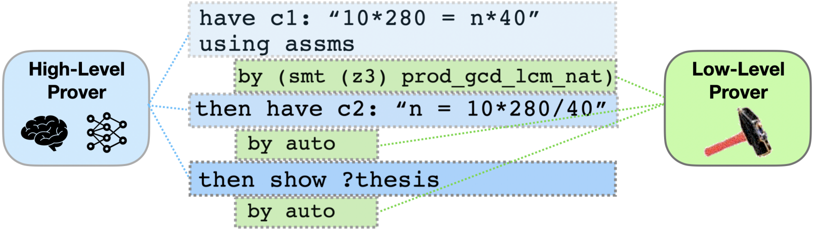
- First-order logic, higher-order logic, SMT

Low-level provers: Sledgehammer



Struggles due to the large search space of possible proofs

Idea: combine high-level and low-level proving



Idea: combine high-level (human, LLM) and low-level proving

Draft, Sketch, Prove: Guiding Formal Theorem Provers with Informal Proofs

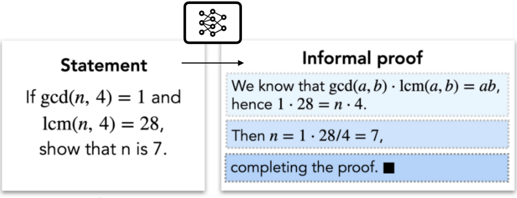
Albert Q. Jiang, Sean Welleck, Jin Peng Zhou
Jiacheng Liu, Wenda Li, Mateja Jamnik
Guillaume Lample, Yuhuai Wu

*ICLR 2023 (**Oral**)*

Draft, Sketch, Prove

Given informal theorem x_I,
formal theorem x_F

1. **Draft** $y_I \sim p(\cdot|x_I)$

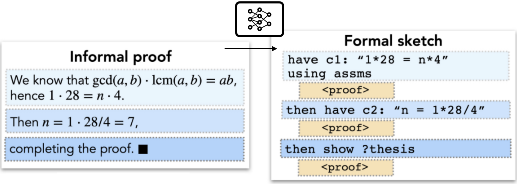


Human-written or LLM-generated draft

Draft, Sketch, Prove

Given informal theorem x_I,
formal theorem x_F

1. Draft $y_I \sim p(\cdot|x_I)$
2. **Sketch** $z_F \sim p(\cdot|x_F, x_I, y_I)$

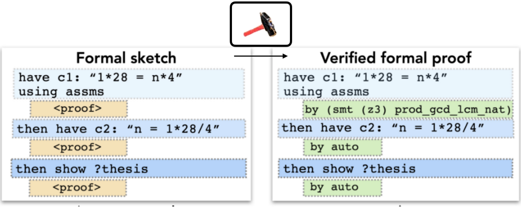


LLM-generated sketch

Draft, Sketch, Prove

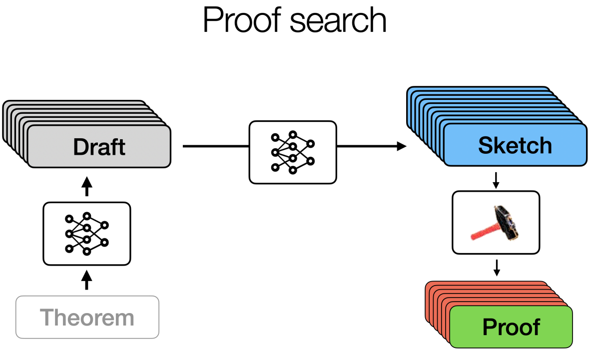
Given informal theorem x_I,
formal theorem x_F

1. Draft $y_I \sim p(\cdot|x_I)$
2. Sketch $z_F \sim p(\cdot|x_F, x_I, y_I)$
3. **Prove** $y_F = f(x_F, z_F)$

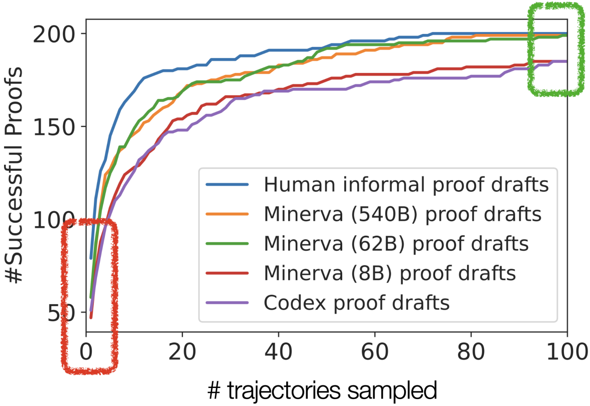


Low-level prover (*Sledgehammer*) fills in the gaps

Draft-sketch-prove

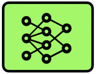


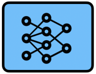
Draft-sketch-prove



Inference-time proof search scaling

Draft-sketch-prove





Statement: Prove that the fraction $\frac{21n+4}{14n+3}$ is irreducible for every natural number n.

Informal Proof (Minerva 62B):

We must show that $\gcd(21n+4, 14n+3) = 1$.
The Euclidean algorithm gives

$$21n+4 = 1 \cdot (14n+3) + 7n + 1$$
$$14n+3 = 2 \cdot (7n+1) + 1.$$

Since $\gcd(7n+1, 1) = 1$, we have $\gcd(21n+4, 14n+3) = 1$.

Formal Proof:

```
theorem imo_1959_p1:
  fixes n :: nat
  shows "gcd (21*n + 4) (14*n + 3) = 1"
proof -
  (* The Euclidean algorithm gives
  21n+4=1\cdot(14n+3)+7n+1
  14n+3=2\cdot(7n+1)+1. *)
  have c0: "21*n + 4 = 1*(14*n + 3) + 7*n + 1"
    by auto   [ATP]
  have c1: "14*n + 3 = 2*(7*n + 1) + 1" using c0
    by auto   [ATP]

  (* Since \gcd(7n+1,1)=1, we have \gcd(21n+4,14n+3)=1. *)
  then have "gcd (7*n + 1) 1 = 1"
    using c1
    by auto   [ATP]
  then have "gcd (21*n + 4) (14*n + 3) = 1"
    using c1
    by (smt (z3) BitM_plus_one ab_semigroup_add_class.add_ac(1)
    add.assoc c0 gcd.commute gcd_add2 gcd_add_mult mult_numeral_1
    numeral_One numeral_eq_Suc numerals(1) semiring_norm(3))   [ATP]
  then show ?thesis
    using c1
    by blast   [ATP]
qed
```

International Math Olympiad problem

Next: building a low-level prover for Lean

Recap:

- Draft-Sketch-Prove: generate high-level sketches and fill in gaps
- Isabelle's *Sledgehammer* calls out to external provers to fill in gaps

Next: building a low-level prover for Lean

Recap:

- Draft-Sketch-Prove: generate high-level sketches and fill in gaps
- Isabelle's *Sledgehammer* calls out to external provers to fill in gaps

Next: can we build a Sledgehammer for Lean?

Premise Selection for a Lean Hammer

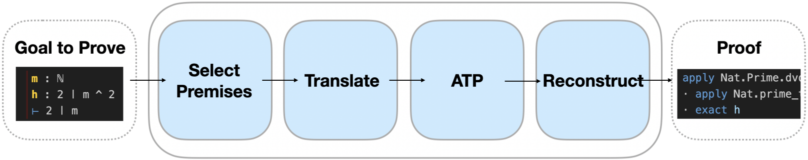
Thomas Zhu, Joshua Clune

Jeremy Avigad, Albert Q. Jiang, Sean Welleck

Under Review 2025

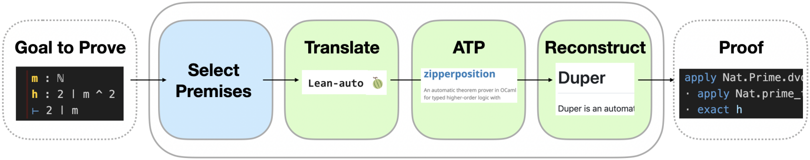
A hammer pipeline

A standard hammer pipeline:



A hammer pipeline

A standard hammer pipeline:

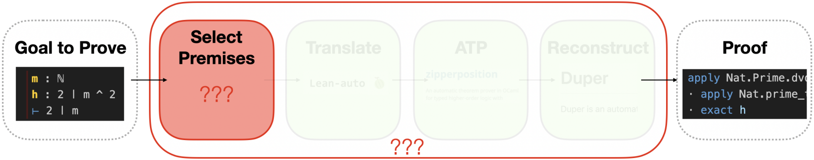


Existing components:

- **Translation**: LeanAuto [Qian et al 2025]
- **ATP**: Zipperposition [Cruanes et al 2015]
- **Reconstruction**: Duper [Clune et al 2024]

A hammer pipeline

A standard hammer pipeline:

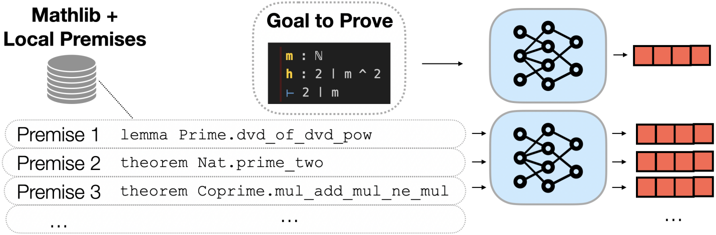


Our challenge:

- **Premise selection**
- **Combine pieces** to create LeanHammer

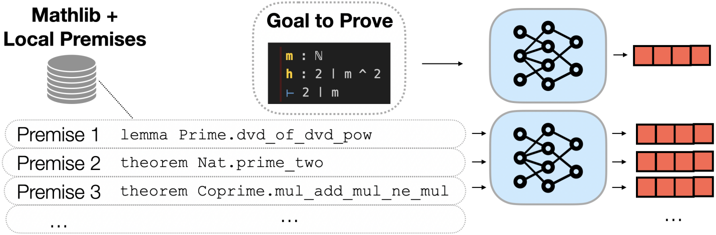
LeanHammer — Neural premise selection

Idea: frame premise selection as retrieval with a neural language model



LeanHammer — Neural premise selection

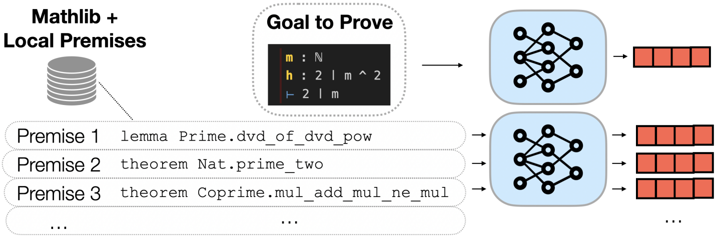
Idea: frame premise selection as retrieval with a neural language model



- Transformer encoder embeds the state and candidate premises

LeanHammer — Neural premise selection

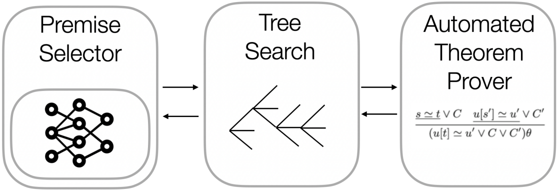
Idea: frame premise selection as retrieval with a neural language model



- Transformer encoder embeds the state and candidate premises
- Contrastive loss on (state, $\{\text{premise}^+\}$, $\{\text{premise}^-\}$) examples
  - Nuance in how to collect and format examples

LeanHammer — Putting it all together

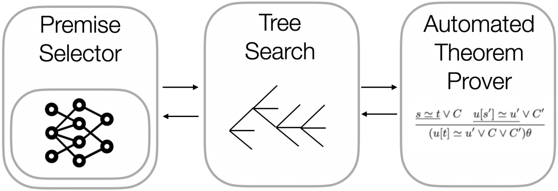
Idea: combine the premise selector and ATP with a tree search



Tree search: *Aesop* [Limperg & From 2023]

LeanHammer — Putting it all together

Idea: combine the premise selector and ATP with a tree search

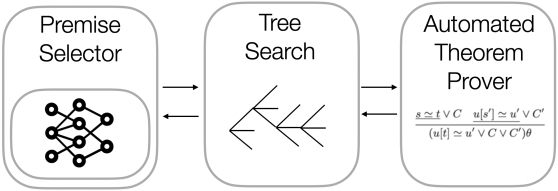


Tree search: *Aesop* [Limperg & From 2023]

1. Queries the automated theorem prover using the premises
2. Applies tactics (e.g. `apply`, `simp_all`) using the premises

LeanHammer — Putting it all together

Idea: combine the premise selector and ATP with a tree search



Tree search: *Aesop* [Limperg & From 2023]

1. Queries the automated theorem prover using the premises
2. Applies tactics (e.g. `apply`, `simp_all`) using the premises

Goes beyond the standard hammer pipeline!

LeanHammer

As a user, simply issue `hammer` at any step of a proof:

```
theorem two_dvd_of_two_dvd_sq {m : ℕ}
  (h : 2 ∣ m ^ 2) : 2 ∣ m := by
  hammer
```

```
Try this:
  apply Nat.Prime.dvd_of_dvd_pow
  · apply Nat.prime_two
  · exact h
```

LeanHammer in action

LeanHammer — Example

Demo: start with human-written proof sketch (from *Mathematics in Lean*)

```
/-- Theorem taken from Mathematics in Lean -/
theorem irrational_sqrt_two {m n : ℕ} (coprime_mn : m.Coprime n) :
    m ^ 2 ≠ 2 * n ^ 2 := by
  intro sqr_eq
  have : 2 ∣ m := by
    sorry
  obtain ⟨k, meq⟩ := dvd_iff_exists_eq_mul_left.mp this
  have : 2 * (2 * k ^ 2) = 2 * n ^ 2 := by
    sorry
  have : 2 * k ^ 2 = n ^ 2 := by
    sorry
  have : 2 ∣ n := by
    sorry
  have : 2 ∣ m.gcd n := by
    sorry
  have : 2 ∣ 1 := by
    sorry
  sorry
```

LeanHammer — Example

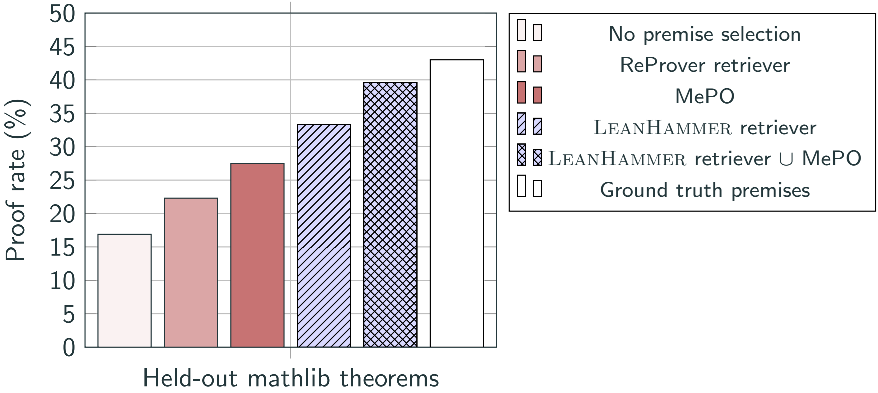
Demo: fill in the gaps (`sorry`s) with LeanHammer

Human Written

```
theorem irrational_sqrt_two {m n : ℕ} (coprime_mn : m.Coprime n) :
    m ^ 2 ≠ 2 * n ^ 2 := by
  intro sqr_eq
  have : 2 ∣ m := by
    apply Lemma.two_dvd_of_two_dvd_sq
    simp_all only [dvd_mul_right]
  obtain ⟨k, meq⟩ := dvd_iff_exists_eq_mul_left.mp this
  have : 2 * (2 * k ^ 2) = 2 * n ^ 2 := by
    subst meq
    simp_all only [dvd_mul_left, mul_eq_mul_left_iff, OfNat.ofNat_ne_zero, or_false]
    (linarith)
  have : 2 * k ^ 2 = n ^ 2 := by
    subst meq
    simp_all only [mul_eq_mul_left_iff, OfNat.ofNat_ne_zero, or_false, dvd_mul_left]
  have : 2 ∣ n := by
    subst meq
    simp_all only [dvd_mul_left]
    (hammerCore [] [*, dvd_mul_right, Nat.modEq_zero_iff_dvd, Nat.ModEq.symm, Nat.modEq_iff_dvd',
        Nat.Coprime.mul_add_mul_ne_mul, frobeniusNumber_pair, add_le_mul, FrobeniusNumber, Nat.ModEq.trans,
        Nat.cast_mul, sq, Nat.Coprime.dvd_of_dvd_mul_left, AddSubmonoid.mem_closure_pair, Nat.chineseRemainder_lt_mul,
        Nat.Coprime.eq_one_of_dvd, Lemma.two_dvd_of_two_dvd_sq] {simpTarget := no_target})
  have : 2 ∣ m.gcd n := by
    clear coprime_mn
    subst meq
    simp_all only [dvd_mul_left]
    apply Nat.dvd_gcd
    · simp_all only [dvd_mul_left]
    · simp_all only
  have : 2 ∣ 1 := by
    subst meq
    simp_all only [dvd_mul_left, Nat.dvd_one, OfNat.ofNat_ne_one]
  subst meq
  simp_all only [Nat.dvd_one, OfNat.ofNat_ne_one]
```

LeanHammer — Quantitative results

Varying the premise selector within LeanHammer:



Recap

Sketching proofs and filling in the gaps

Recap

Sketching proofs and filling in the gaps

- **Draft-Sketch-Prove (DSP)**

Recap

Sketching proofs and filling in the gaps

- **Draft-Sketch-Prove (DSP)**
- **LeanHammer**

This talk: Bridging Informal and Formal

1. Informal thoughts
2. Informal provers
3. **Towards research-level mathematics**

III: Research-level mathematics

What does it look like to formalize research-level math?[3]

arXiv > math > arXiv:2311.05762

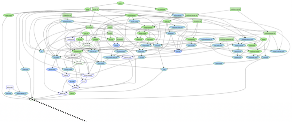
Mathematics > Number Theory

[Submitted on 9 Nov 2023 (v1), last revised 12 Dec 2023 (this version, v2)]

On a conjecture of Marton

W. T. Gowers, Ben Green, Freddie Manners, Terence Tao

arXiv



Theorem 7.2 (PFR)

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

Proof ▸

Let U_A be the uniform distribution on A (which exists by Lemma 2.5), thus $H[U_A] = \log|A|$ by Lemma 2.7. By Lemma 2.3 and the fact that $U_A + U_A$ is supported on $A + A$, $H[U_A + U_A] \leq \log|A+A|$. By Definition 3.7, the doubling condition $|A+A| \leq K|A|$ therefore gives

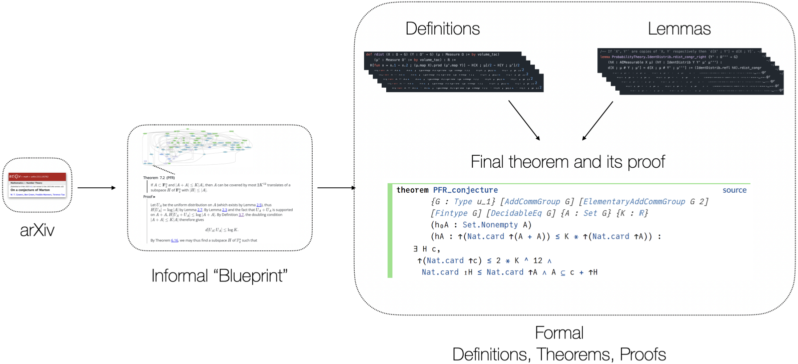
$$d[U_A; U_A] \leq \log K.$$

By Theorem 6.16, we may thus find a subspace H of $\mathbb{F}_2^n$ such that

Informal "Blueprint"

[3] *Formalizing the proof of PFR in Lean4 using Blueprint: a short tour* by Terence Tao

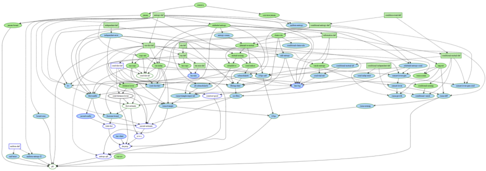
What does it look like to formalize research-level math?[3]



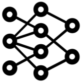
[3] *Formalizing the proof of PFR in Lean4 using Blueprint: a short tour* by Terence Tao

Where can AI help?

As a start, can AI help with filling in small parts of the blueprint?

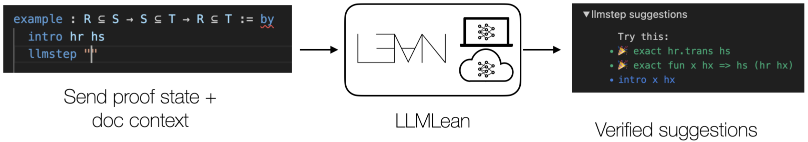


```
/-- `H[X | Y=y] = ∑_s P[X=s | Y=y] log 1/(P[X=s | Y=y])`. -/
lemma entropy_cond_eq_sum (μ : Measure Ω) (y : T) :
    H[X | Y ← y ; μ] = ∑' x, negMulLog ((μ[|Y ← y]).map X {x}).toReal := by
```



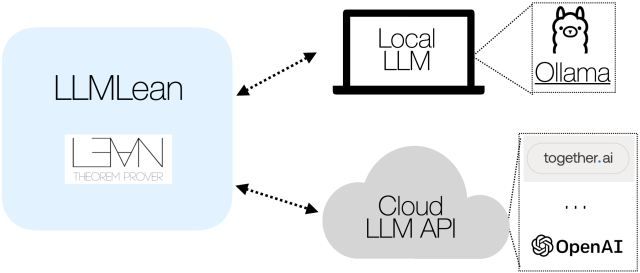
???

Where can AI help? — Existing tools



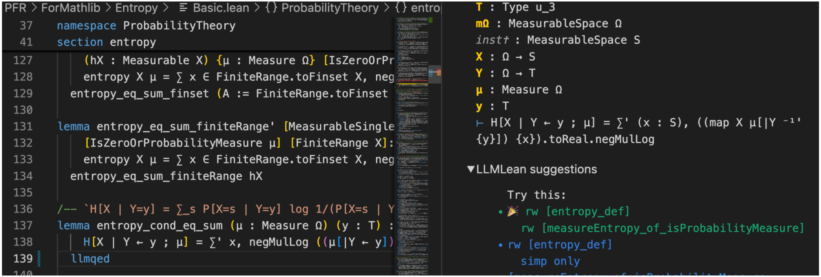
LLMLean: https://github.com/cmu-l3/llmlean

Where can AI help? — Existing tools



LLMLean: https://github.com/cmu-l3/llmlean

Where can AI help? — Existing tools



LLMLean example on Polynomial Freiman Rusza Conjecture project

Where can AI help? — Benchmarking gap

Math competition problems

Theorem to prove

```
theorem imo_1964_p1_2 (n : ℕ) : ¬7 ∣ 2 ^ n + 1 := by
  sorry
```

- Self-contained
- Uses standard results

Where can AI help? — Benchmarking gap

Math competition problems

Theorem to prove

```
theorem imo_1964_p1_2 (n : ℕ) : ¬7 ∣ 2 ^ n + 1 := by
  sorry
```

- Self-contained
- Uses standard results

Real projects

Definitions

Lemmas

Theorem to prove

```
/-- `H[X | Y=y] = ∑_s P[X=s | Y=y] log 1/(P[X=s | Y=y])`. -/
lemma entropy_cond_eq_sum (μ : Measure Ω) (y : T) :
    H[X | Y ← y ; μ] = ∑' x, negMulLog ((μ[|Y ← y]).map X {x}).toReal := by
```

- Part of a project
- Uses new definitions and lemmas

Where can AI help? — Benchmarking gap

Math competition problems

Theorem to prove

```
theorem imo_1964_p1_2 (n : ℕ) : ¬7 ∣ 2 ^ n + 1 := by
  sorry
```

Real projects

Definitions

Lemmas

Theorem to prove

```
/-- `H[X | Y=y] = ∑_s P[X=s | Y=y] log 1/(P[X=s | Y=y])`. -/
lemma entropy_cond_eq_sum (μ : Measure Ω) (y : T) :
    H[X | Y ← y ; μ] = ∑' x, negMulLog ((μ[|Y ← y]).map X {x}).toReal := by
```

- Self-contained
- Uses standard results

- Part of a project
- Uses new definitions and lemmas
- New Benchmark: **miniCTX**

miniCTX: Neural Theorem Proving with (Long-)Contexts
Jiewen Hu, Thomas Zhu, Sean Welleck. *ICLR 2025 (**Oral**)*.

This talk: Bridging Informal and Formal

1. Informal thoughts
    - Training models to think informally
        - Lean-STaR
2. Informal provers
    - Sketching proofs and filling in the gaps
        - Draft, Sketch, Prove
        - LeanHammer
3. Towards research-level mathematics
    - Assisting in research-level projects
    - Practical tools
    - MiniCTX

Thank you!

Collaborators on works in this talk (alphabetical by last name):

- Jeremy Avigad (CMU)
- Joshua Clune (CMU)
- Jiewen Hu (CMU)
- Mateja Jamnik (Cambridge)
- Albert Q. Jiang (Cambridge, Mistral)
- Timothee Lacroix (Meta, Mistral)
- Guillaume Lample (Meta, Mistral)
- Haohan Lin (Tsinghua)
- Wenda Li (Edinburgh)
- Jiacheng Liu (Washington)
- Zhiqing Sun (CMU, OpenAI)
- Yuhuai (Tony) Wu (Google, X.ai)
- Yiming Yang (CMU)
- Jin Peng Zhou (Cornell)
- Thomas Zhu (CMU)

Sean Welleck
CMU School of Computer Science
Learning, Language, and Logic (L3) Lab
www.wellecks.com
wellecks@cmu.edu