

Mathematics as code

Sean Welleck

April 4, 2024

Carnegie Mellon University

LLMs in expert domains

- Finance
- Medicine
- [Mathematics](#)

¹See, e.g. Harvard New Technologies in Mathematics: 1/2021 (Christian Szegedy), 3/2022 (Stan Polu), 10/2022 (Guy Gur-Ari), 10/2023 (Alex Gu), 12/2023 (Katherine Collins), ...;

LLMs in expert domains

- Finance
- Medicine
- **Mathematics**
 - Solve problems
 - Open-ended dialogue
 - Help generate verified proofs
 - ...

¹See, e.g. Harvard New Technologies in Mathematics: 1/2021 (Christian Szegedy), 3/2022 (Stan Polu), 10/2022 (Guy Gur-Ari), 10/2023 (Alex Gu), 12/2023 (Katherine Collins), ...;

“Informal” mathematics

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

- Flexible
- Useful (e.g., in education)
- No correctness guarantees

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.

Figure 1: Language model solution.

Math as source code

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

$$1 + 1 = 2$$

proof ✓

```
lemma one_plus_one_equals_two:
  .. shows "1 + 1 = 2"
proof -
  have "1 + 1 = Suc (0 + 1)" by simp
  also have "... = Suc 1" by simp
  ..
  also have "... = 2" by simp
  ..
  finally show ?thesis by simp
qed
```

Figure 2: Math as source code.

Math as **source code**

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

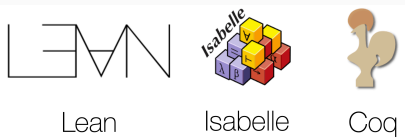


Figure 3: Theorem proving languages

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



How is formal math used in practice?

Growing use in mathematics:

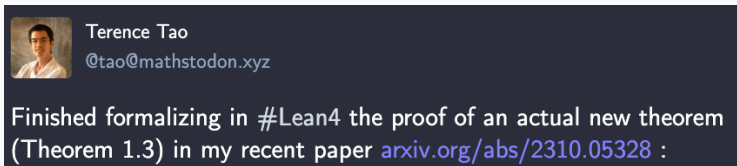


Figure 4: Terence Tao's Lean formalization project (October 2023)

How is formal math used in practice?

Growing use in mathematics:

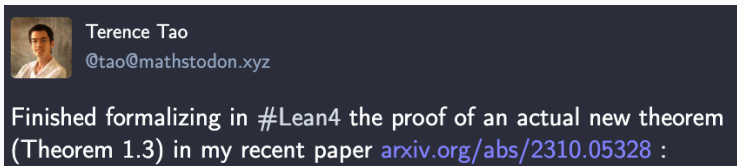


Figure 4: Terence Tao's Lean formalization project (October 2023)

- **Lean Mathlib** project: 1+ million lines of code, 300+ contributors
- **Courses** at CMU, Imperial College London, Fordham, JHU, ...
- **New journal (2024):** *Annals of Formalized Mathematics*

How is formal math used in practice?

Why?¹

- Collaboration
- Instant feedback
- Correctness guarantees
- ...

¹See e.g., *Mathematics and the formal turn*, AFM Aims and Scope

Why is AI \cap formal math important?

LLMs for formal math

- Automate proofs
- Translate informal to formal
- Suggest strategies
- ...

Why is AI \cap formal math important?

Formal math for LLMs

- **Verifiable**
 - Prevent incorrect math and code generation
 - Feedback signal for learning

Why is AI \cap formal math important?

Formal math for LLMs

- **Verifiable**
 - Prevent incorrect math and code generation
 - Feedback signal for learning
- Tests **reasoning**
 - From easy: $1+1 = 2$
 - To hard: Fermat's Last Theorem

Why is AI \cap formal math important?

Formal math for LLMs

- **Verifiable**
 - Prevent incorrect math and code generation
 - Feedback signal for learning
- Tests **reasoning**
 - From easy: $1+1 = 2$
 - To hard: Fermat's Last Theorem
- Complementary **tools**
 - Computer algebra, SAT/SMT solvers, ...
 - Rule-based automation, ...

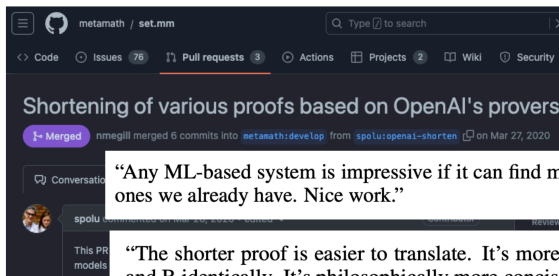
And still far from being “solved” (April 2024)...

Generative Language Modeling for Automated Theorem Proving

Stanislas Polu
OpenAI
spolu@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

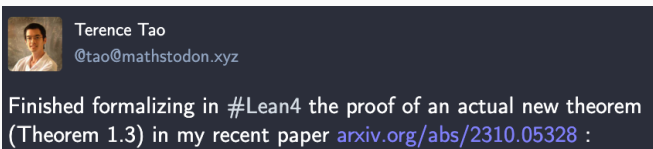
Figure 5: *gpt-f* (2020)



“Any ML-based system is impressive if it can find many shorter proofs than the ones we already have. Nice work.”

“The shorter proof is easier to translate. It’s more symmetric in that it treats A and B identically. It’s philosophically more concise in that it doesn’t rely on the existence of a universal class of all sets.”

Figure 6: *gpt-f* (2020)



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.

Figure 7: Terence Tao's Lean formalization project (October 2023)

1. Intro: Foundation models \cap mathematics

- Informal and formal mathematics
- Why is formal mathematics important?

2. Part I: Build a LLM formal theorem proving tool

- Data, training, proof search, evaluation, tool

3. Part II: Leveraging *informal* mathematical data

- Via foundation model
- Via translation

1. Intro: Foundation models \cap mathematics
 - Informal and formal mathematics
 - Why is formal mathematics important?
2. **Part I: Build a LLM formal theorem proving tool**
 - Data, training, proof search, evaluation, tool
3. Part II: Leveraging *informal* mathematical data
 - Via foundation model
 - Via translation

PART I:

Build a [L]LM proving tool

Build a [L]LM proving tool²

Topic	Notebook
0. Intro	notebook
1. Data	notebook
2. Learning	notebook
3. Proof Search	notebook
4. Evaluation	notebook
5. Context	notebook
6. LLMLean tool	notebook

Interactive notebooks and code: github.com/cmu-l3/ntptutorial-II

²Update of *Tutorial on neural theorem proving*, IJCAI 2023, github.com/wellecks/ntptutorial

Build a [L]LM proving tool³

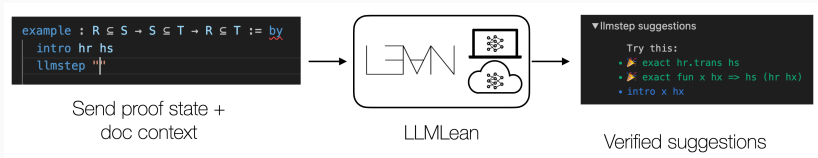
Artifacts:

Name	Huggingface
Data: mathlib extractions	l3lab/ntp-mathlib
Data: instructions (state-tactic)	l3lab/ntp-mathlib-instruct-st
Data: instructions (+context)	l3lab/ntp-mathlib-instruct-ctx
Model: state-tactic	l3lab/ntp-mathlib-st-deepseek-coder-1.3b
Model: +context	l3lab/ntp-mathlib-context-deepseek-coder-1.3b

Datasets and models on Huggingface: <https://huggingface.co/l3lab>

³Update of *Tutorial on neural theorem proving*, IJCAI 2023, github.com/wellecks/ntptutorial

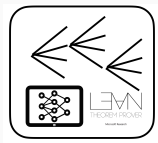
Build a [L]LM proving tool⁴



⁴Update of *Tutorial on neural theorem proving*, IJCAI 2023, github.com/wellecks/ntptutorial

Next-step (tactic) prediction

- Language model suggests next-proof-steps
- Generate a full proof via tree search



¹E.g., [Polu & Sutskever 2020], [Han et al 2021], [Jiang et al 2022], [Yang et al 2023]

- Model: $p_{\theta}(\mathbf{y}|\mathbf{x}; \mathcal{D})$
 - \mathbf{y} : output sequence
 - \mathbf{x} : input sequence
 - θ : parameters (e.g., transformer)
 - \mathcal{D} : dataset

- Model: $p_{\theta}(\mathbf{y}|\mathbf{x}; \mathcal{D})$
 - \mathbf{y} : output sequence
 - \mathbf{x} : input sequence
 - θ : parameters (e.g., transformer)
 - \mathcal{D} : dataset
- Learning:
 - $\arg \max_{\theta} \sum_{\mathbf{y} \in \mathcal{D}} \log p_{\theta}(\mathbf{y})$

- Model: $p_{\theta}(\mathbf{y}|\mathbf{x}; \mathcal{D})$
 - \mathbf{y} : output sequence
 - \mathbf{x} : input sequence
 - θ : parameters (e.g., transformer)
 - \mathcal{D} : dataset
- Learning:
 - $\arg \max_{\theta} \sum_{\mathbf{y} \in \mathcal{D}} \log p_{\theta}(\mathbf{y})$

- Model: $p_{\theta}(\mathbf{y}|\mathbf{x}; \mathcal{D})$
 - \mathbf{y} : output sequence
 - \mathbf{x} : input sequence
 - θ : parameters (e.g., transformer)
 - \mathcal{D} : dataset
- Learning:
 - $\arg \max_{\theta} \sum_{\mathbf{y} \in \mathcal{D}} \log p_{\theta}(\mathbf{y})$
- Inference:
 - $\mathbf{y} = f(p_{\theta}(\cdot|\mathbf{x}))$
 - f : e.g., sampling

Problem setup

Proof: sequence of (state, step)

- $(x_0, y_0), \dots, (x_T, y_T)$
- x_t : proof state from Lean
- y_t : proof step (“tactic”)



The screenshot shows the Lean theorem prover interface. At the top, a code editor contains the definition: `example : R ⊆ S → S ⊆ T → R ⊆ T := by`. Below this, a "Tactic state" window is open, displaying the goal: `1 goal`, the type `α : Type`, and the set `R S T : Set α`. The goal is `⊢ R ⊆ S → S ⊆ T → R ⊆ T`. To the right of the tactic state is the Lean logo and the text "THEOREM PROVER" and "Microsoft Research". Below the tactic state, another code editor shows the same definition with the tactic `intro h1 h2` applied, resulting in `intro h1 h2` being entered after the `by` keyword.

Problem setup

Proof: sequence of (state, step)

- $(x_0, y_0), \dots, (x_T, y_T)$
- x_t : proof state from Lean
- y_t : proof step (“tactic”)

The image shows a screenshot of the Lean theorem prover interface. On the left, the text x_{t+1} is written, with a curved arrow pointing to the 'Tactic state' window. The 'Tactic state' window displays the following text:

```
▼ Tactic state
1 goal
α : Type
R S T : Set α
h₁ : R ⊆ S
h₂ : S ⊆ T
⊢ R ⊆ T
```

To the right of the 'Tactic state' window is the Lean logo, which reads 'LEAN THEOREM PROVER' with 'Microsoft Research' underneath. Below the 'Tactic state' window, the text y_{t+1} is written, with a curved arrow pointing to the code editor. The code editor shows the following code:


```
example : R ⊆ S → S ⊆ T → R ⊆ T := by
  intro h₁ h₂
  exact h₁.trans h₂
```

At the bottom of the screenshot, there is a black bar with the text 'No goals' in blue, followed by a colorful party horn icon.

Problem setup

Proof: sequence of (state, step)

- $(x_0, y_0), \dots, (x_T, y_T)$
- x_t : proof state from Lean
- y_t : proof step (“tactic”)



The screenshot shows the Lean theorem prover interface. On the left, a small icon of a neural network is connected by arrows to the 'Tactic state' and the proof code. The 'Tactic state' panel displays:

```
▼ Tactic state
1 goal
α : Type
R S T : Set α
h₁ : R ⊆ S
h₂ : S ⊆ T
⊢ R ⊆ T
```

To the right of the tactic state is the Lean logo and the text 'THEOREM PROVER' and 'Microsoft Research'. Below the tactic state, the proof code is shown:

```
example : R ⊆ S → S ⊆ T → R ⊆ T := by
  intro h₁ h₂
  exact h₁.trans h₂
```

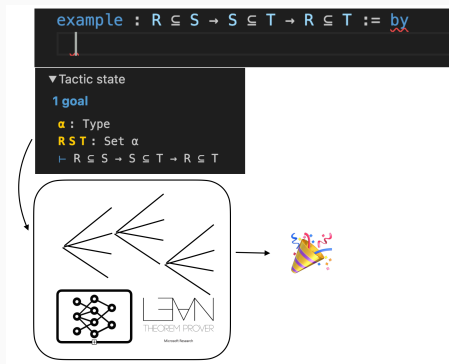
At the bottom, a 'No goals' message is displayed next to a colorful confetti icon.

Idea: train language model $p_\theta(y_t|x_t)$ on a dataset of (state, step) pairs

Problem setup

Proof: sequence of (state, step)

- $(x_0, y_0), \dots, (x_T, y_T)$
- x_t : proof state from Lean
- y_t : proof step (“tactic”)



...then use the model + tree search to prove full theorems

1. Data



- Extract (state, tactic) pairs from Lean projects. Tools:
 - **ntp-training-data** (this tutorial)⁵
 - Lean Dojo [2]

⁵Based on github.com/semorrison/lean-training-data

1. Data (ntp-training-data)

- Format each (state, tactic) pair as (prompt, completion) example:

```
└─ You are proving a theorem in Lean 4.  
You are given the following information:  
- The current proof state, inside [STATE]...[/STATE]  
  
Your task is to generate the next tactic in the proof.  
Put the next tactic inside [TAC]...[/TAC]  
-/  
[STATE]  
m n : ℕ  
h : Nat.Coprime m n  
⊢ Nat.gcd m n = 1  
[/STATE]  
[TAC]
```

Prompt:

Completion:

```
rw [Nat.Coprime] at h  
[/TAC]
```

Mathlib gives a dataset of $\approx 300,000$ examples

1. Data (ntp-training-data)

The screenshot shows the Hugging Face dataset viewer interface for the dataset 'l3lab/ntp-mathlib-instruct-st'. The page includes a search bar, navigation tabs for 'Dataset card', 'Viewer', 'Files and versions', 'Community', and 'Settings'. The 'Dataset Viewer' section shows the dataset is split into 'train' with 291k rows. A search bar is provided to search within the dataset. Below this, there are four columns: 'task', 'prompt', 'completion', and 'metadata'. Each column has a small histogram showing the distribution of values. The 'task' column shows a single value 'tactic_prediction'. The 'prompt' column shows a sample prompt: '/- You are proving a theorem in Lean 4. You are given th...'. The 'completion' column shows a sample completion: 'apply G.mk_eq [/TAC]'. The 'metadata' column shows a dictionary: {'task': 'tactic_predi', 'Examples/Mathlib', 'fi'.

task	prompt	completion	metadata
string · classes 1 value	string · lengths 259 159k	string · lengths 10 9.01k	dict
tactic_prediction	/- You are proving a theorem in Lean 4. You are given th...	apply G.mk_eq [/TAC]	{ "task": "tactic_predi", "Examples/Mathlib", "fi

Figure 8: Data on Huggingface

[NOTEBOOK DEMO]

2. Learning

- Standard supervised learning on \mathcal{D} :

$$\arg \max_{\theta} \sum_{(x_t, y_t) \in \mathcal{D}} \log p_{\theta}(y_t | x_t)$$

Learning (ntp-tune)

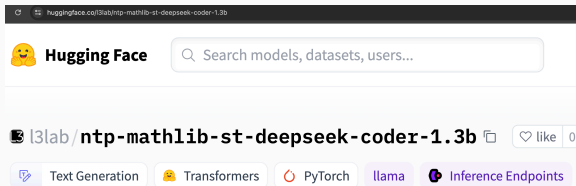
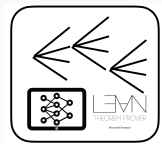


Figure 9: Trained tutorial model on Huggingface

[\[NOTEBOOK DEMO\]](#)

3. Proof search

- Use generator $p_{\theta}(y_t|x_t)$ to generate a full proof y_1, \dots, y_T
- Standard approach: *Best-first search*



3. Proof search | Best-first search

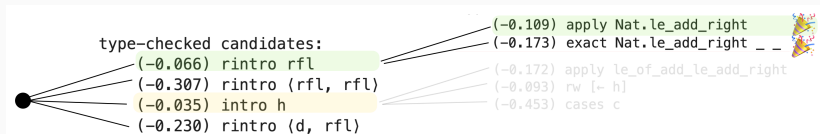


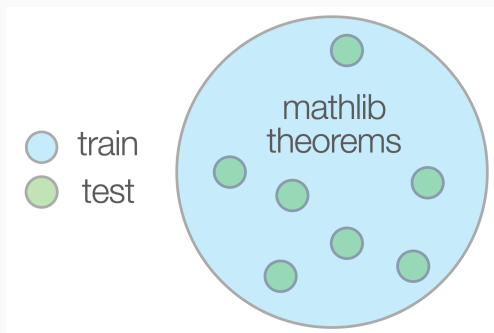
Figure 10: Best-first search⁶

⁶Example scoring function $\frac{1}{2} \sum_t \log p_\theta(y_t|x_t)$

[NOTEBOOK DEMO]

4. Evaluation

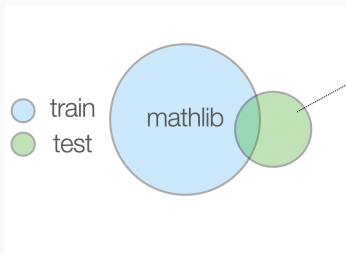
- Proof search on held-out theorems from the training distribution



4. Evaluation | benchmarks

Benchmarks evaluate problems drawn from a different distribution:

- **miniF2F** [3]: 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 :=
begin
```

4. Evaluation

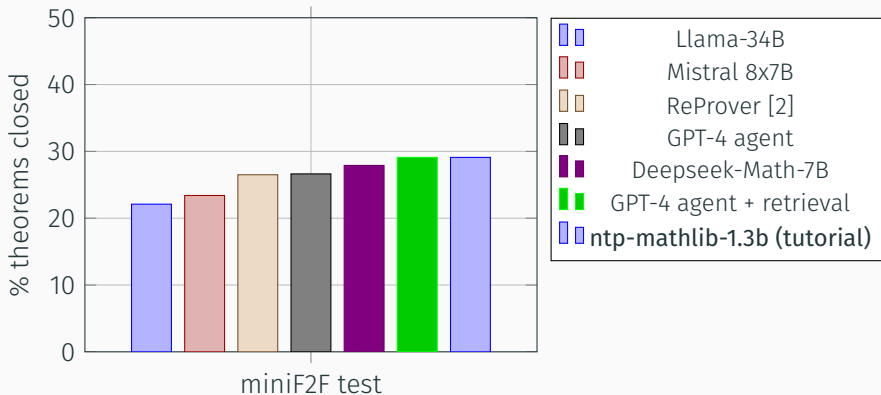


Figure 11: Proof search performance on miniF2F theorems. The model we trained in the tutorial notebooks gets 29.1% (71/244) on miniF2F test.

4. Evaluation

```
-- from mathlib:
theorem prod_mono
  {s1 s2 : Subsemiring R} (hs : s1 ≤ s2)
  {t1 t2 : Subsemiring S} (ht : t1 ≤ t2) :
  s1.prod t1 ≤ s2.prod t2 := by
  intro x hx
  simp_rw [Subsemiring.mem_prod]
  cases' x with x_fst x_snd
  exact ⟨hs hx.1, ht hx.2⟩

-- from miniF2F:
theorem mathd_algebra_159 (b : ℝ) (f : ℝ → ℝ)
  (h0 : ∀ x, f x = 3*x^4 - 7*x^3 + 2*x^2 - b*x + 1)
  (h1 : f 1 = 1) : b = -2 := by
  apply eq_neg_of_add_eq_zero_left
  rw [h0] at h1
  norm_num at h1
  linarith
```

Figure 12: Generated proofs

[NOTEBOOK DEMO]

5. Extensions: context

Real theorem proving uses *context* (e.g., new definitions, lemmas) [1]:

```
variable { $\Omega$  : Type*}[Fintype  $\Omega$ ]  
  
structure my_object ( $\Omega$  : Type*)[Fintype  $\Omega$ ] :=  
  (f :  $\Omega \rightarrow \mathbb{R}$ )  
  (cool_property :  $\forall x : \Omega, 0 \leq f x$ )  
  
theorem my_object_sum_nonneg (o1 o2: my_object  $\Omega$ ) : o1.f + o2.f  $\geq 0$  := by  
  apply add_nonneg  
  · apply o1.cool_property  
  · apply o2.cool_property
```

Figure 13: The theorem uses a newly defined *my_object* and *cool_property* [1].

A model trained on (state, tactic) examples does not access context outside of its training set.

5. Extensions: context

Train a context-dependent model:

$$p_{\theta}(y_t|x_t, c_t), \tag{1}$$

e.g., where c is the preceding file contents.

5. Extensions: context

```
└─ You are proving a theorem in Lean 4.
You are given the following information:
- The file contents up to the current tactic, inside [CTX]...[/CTX]
- The current proof state, inside [STATE]...[/STATE]

Your task is to generate the next tactic in the proof.
Put the next tactic inside [TAC]...[/TAC]
-/
[CTX]
import Mathlib.Data.Nat.Prime

theorem test_thm (m n : Nat) (h : m.Coprime n) : m.gcd n = 1 := by

[/CTX]
[STATE]
m n : ℕ
h : Nat.Coprime m n
├ Nat.gcd m n = 1
[/STATE]
[TAC]
```

Prompt:

Completion:

```
rw [Nat.Coprime] at h
[/TAC]
```

[NOTEBOOK DEMO]

6. Integrating LLMs and Lean

LLMLEAN: tools that integrate LLMs into the Lean proof assistant

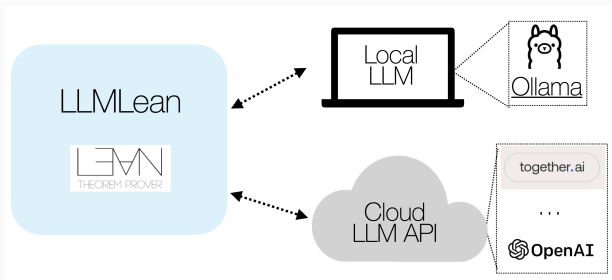


Figure 14: github.com/cmu-l3/llmlean

6. Integrating LLMs and Lean

Example: Verified *llmstep*⁷ suggestions on a MacBook Pro:

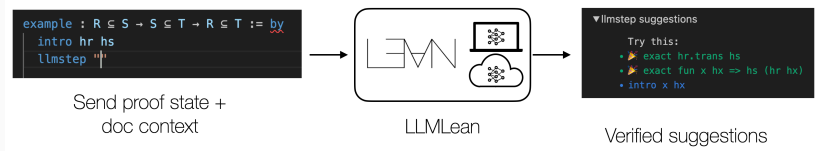


Figure 15: github.com/cmu-l3/llmlean

[DEMO]

⁷*LLMstep: LLM proofstep suggestions in Lean* [Welleck & Saha 2023]

- Train next-tactic generators, $p_{\theta}(y_t|x_t, c_t)$
- Prove theorems with a best-first tree search
- Data, Learning, Search, Evaluation, LLMLEAN tool

Check out the notebooks, code, data, and models!

- Notebooks and code: github.com/cmu-l3/ntptutorial-II
 - Data: NTP-TRAINING-DATA
 - Proof search: NTP-INTERACT
 - Fine-tuning: NTP-TUNE
- Datasets and models:
 - <https://huggingface.co/l3lab>

Extensions (formal-to-formal tactic generation)

Active research area with many extensions and related works:

Reinforcement learning

- Expert Iteration [Polu et al 2022]
- Thor with Expert Iteration [Wu et al 2022]

Search algorithms

- Hypertree Proof Search [Lample et al 2022]
- DT-Solver [Wang et al 2023]

Retrieval

- Repeater [Yang et al 2023]

Integrating symbolic provers

- Thor [Jiang et al 2022]

LLM agents

- COPRA [Thakur et al 2024]

Benchmarks

- MINIF2F [Zheng et al 2021]
- PROOFNET [Azerbaiyev et al 2023]

Data extraction/interaction

- PISA [Jiang et al 2021] (Isabelle)
- PACT [Han et al 2021] (Lean 3)
- LEAN-TRAINING-DATA [Morrison 2023]
- NTP-TRAINING-DATA (this tutorial)
- Lean Dojo [Yang et al 2023]

Tools

- LLMLEAN/LLMSTEP [Welleck & Saha 2023]
- Lean Copilot [Song et al 2023]

Non-exhaustive list; also more extensions in the next section!

1. Intro: LLMs \cap mathematics
 - Informal and formal mathematics
 - Why is formal mathematics important?
2. Part I: Build a LLM formal theorem proving tool
 - Data, training, proof search, evaluation, tool
3. **Part II: Leveraging *informal* mathematical data**
 - Via foundation model
 - Via translation and guidance

PART II: Leveraging *informal* mathematical data

Leveraging *informal* mathematical data

- *Previous*: train a model purely on formal data

Leveraging *informal* mathematical data

- *Previous*: train a model purely on formal data
- *Next*: use *informal* mathematical data
 - Latex proofs
 - Math textbooks, papers, websites, ...
 - Conversations, ...

Why leverage informal data?

1. Data scarcity

- Lean: 300 million tokens
- Arxiv: 29 billion tokens
- General web data: > 5 trillion tokens

Why leverage informal data?

1. Data scarcity

- Lean: 300 million tokens
- Arxiv: 29 billion tokens
- General web data: > 5 trillion tokens

2. Guiding search

- Informal proofs can help cut down the space of possible proofs

Why leverage informal data?

1. Data scarcity

- Lean: 300 million tokens
- Arxiv: 29 billion tokens
- General web data: > 5 trillion tokens

→ *transfer knowledge by adapting a generalist model to mathematical data*

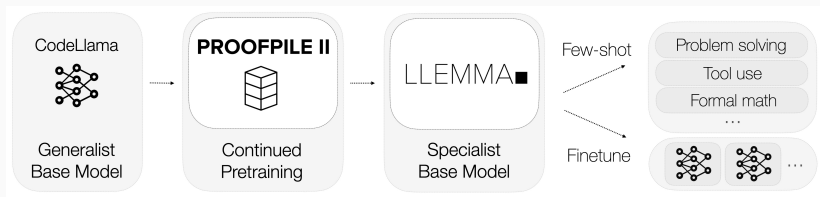
2. Guiding search

- Informal proofs can help cut down the space of possible proofs

Foundation model for mathematics : LLEMMA⁸

Recipe for adapting a language model to mathematical data:

- Collect high-quality, diverse math-related corpus, PROOFPILE II
- Continue pretraining a base model (e.g., Code Llama) on PROOFPILE II



⁸*Llemma: an open language model for mathematics*

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, Sean Welleck

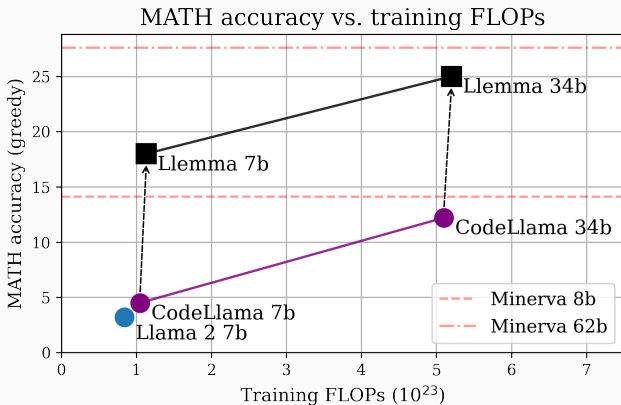
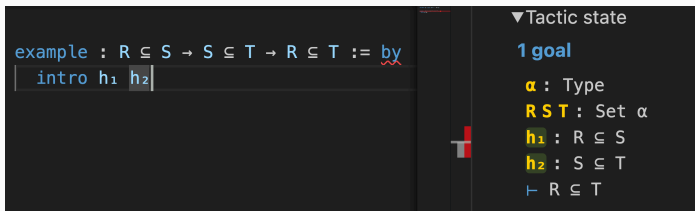


Figure 16: LLEMMA improves with a modest amount of math-specific compute

Proofpile II : code + web data + Arxiv papers

- ALGEBRAICSTACK – 11B tokens from 17 programming languages
 - 1.5B tokens of formal math
 - Extracted Lean and Isabelle goal states



The image shows a screenshot of a Lean proof editor. On the left, the code is displayed in a dark theme with syntax highlighting. The code defines an example and starts a proof with the `intro` tactic. On the right, the tactic state is shown, listing the current goal and the hypotheses available in the context.

```
example : R ⊆ S → S ⊆ T → R ⊆ T := by
  intro h1 h2
```

▼Tactic state

1 goal

- α : Type
- $R S T$: Set α
- h_1 : $R \subseteq S$
- h_2 : $S \subseteq T$
- $\vdash R \subseteq T$

Figure 17: Lean code (left) and goal state (right)

- 14.7 billion tokens of math-related web data

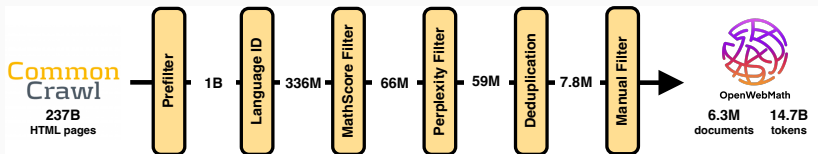


Figure 18: OpenWebMath pipeline.

⁹*OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text.*
Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, Jimmy Ba

Traditional proof search: $p_{\theta}(\text{next-tactic}|\text{state})$ + best-first search.

- We implement a *few-shot* version by providing LLEMMA with 3 (*state, next-tactic*) examples in its prompt

LLEMMA formal theorem proving

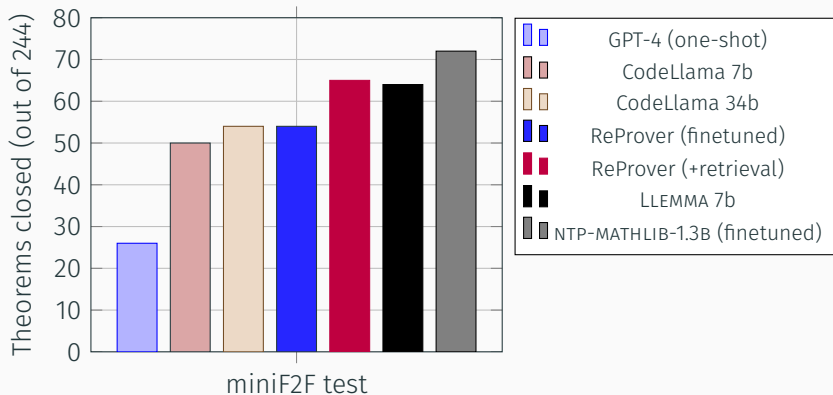
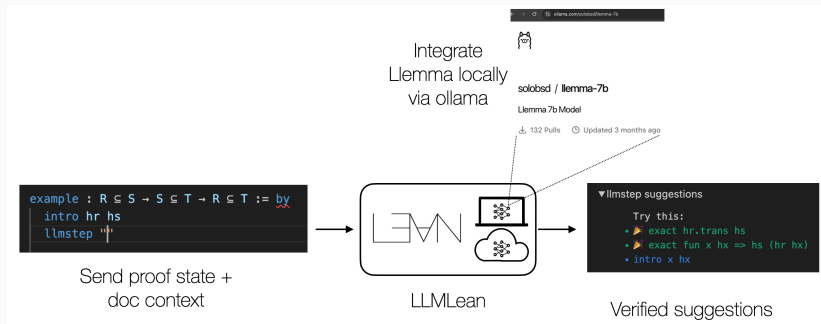


Figure 19: Few-shot proving in Lean with LLEMMA

LLEMMA on your laptop with LLMLEAN¹⁰



DEMO

¹⁰<https://github.com/cmu-l3/llmlean>, based on *LLMstep: LLM proofstep suggestions in Lean*. Sean Welleck & Rahul Saha, Neurips Math+AI 2023

- Leverage informal data by pretraining + adaptation to diverse mathematical data
 - LLEMMA: 7B and 34B CodeLLama further trained on PROOFPILE II
- Open platform for research:
 - Code/Models/Data: <https://github.com/EleutherAI/math-lm>

Why leverage informal data?

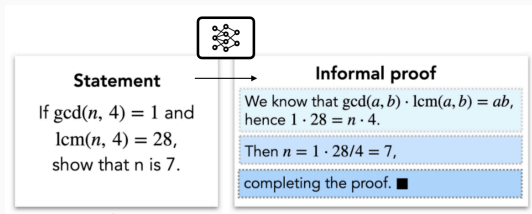
1. Data scarcity

2. **Guiding search**

- Informal proofs can help cut down the space of possible proofs

Given informal theorem x_I ,
formal theorem x_F

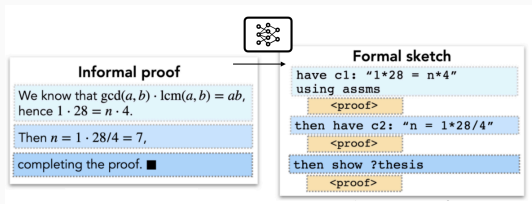
1. Draft $y_I \sim p(\cdot|x_I)$



¹¹*Draft, Sketch, Prove: Guiding Formal Theorem Provers with Informal Proofs*
Jiang, Welleck, Zhou, Lacroix, Liu, Li, Jamnik, Lample, Wu. ICLR 2023

Given informal theorem x_I ,
formal theorem x_F

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



¹¹*Draft, Sketch, Prove: Guiding Formal Theorem Provers with Informal Proofs*
Jiang, Welleck, Zhou, Lacroix, Liu, Li, Jamnik, Lample, Wu. ICLR 2023

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** $y_F = f(x_F, z_F)$

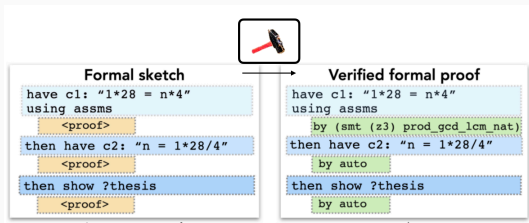


Figure 20: "Classical" prover *Sledgehammer*

¹¹*Draft, Sketch, Prove: Guiding Formal Theorem Provers with Informal Proofs*
Jiang, Welleck, Zhou, Lacroix, Liu, Li, Jamnik, Lample, Wu. ICLR 2023

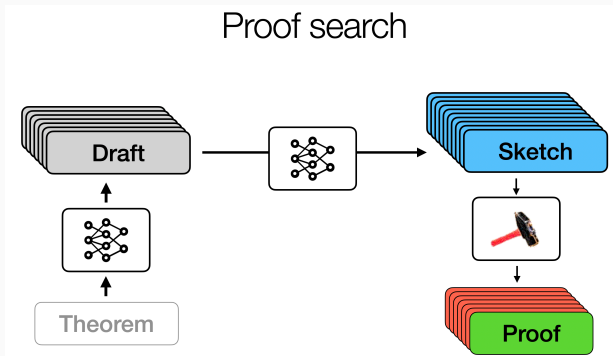


Figure 21: Proof search

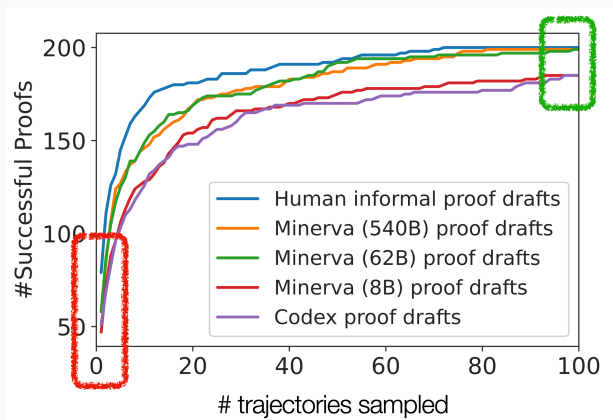


Figure 22: Proof search scaling



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
```

Figure 23: International Math Olympiad problem

[DEMO NOTEBOOK]

Why leverage informal data?

1. Data scarcity

- Adapt a foundation model to “all” mathematical data

2. Guiding search

- Informal proofs can help cut down the space of possible proofs

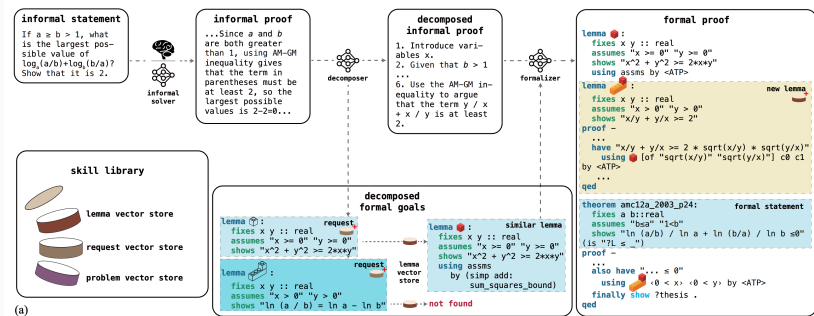


Figure 24: Lego Prover

¹²*Lego-Prover: Neural Theorem Proving with Growing Libraries*

Success rate	LLM	miniF2F-valid	miniF2F-test
<i>Baselines</i>			
Thor (Jiang et al., 2022a)	-	28.3%	29.9%
Thor + expert iteration (Wu et al., 2022)	Codex	37.3%	35.2%
Draft, sketch, and Prove (Jiang et al., 2022b)	Codex	42.6%	39.3%
Subgoal-Learning (Zhao et al., 2023)	ChatGPT	48.0%	45.5%
<i>Ours (100 attempts)</i>			
LEGO-Prover (model informal proof)	ChatGPT	52.0%	45.5%
LEGO-Prover (human informal proof)	ChatGPT	55.3%	50.0%
LEGO-Prover*	ChatGPT	57.0%	50.0%

Figure 25: Lego Prover results

¹²*Lego-Prover: Neural Theorem Proving with Growing Libraries*

Haiming Wang, Huajian Xin, Chuanyang Zheng, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, Xiaodan Liang, ICLR 2024 (Oral)

Extensions and related (leveraging informal data)

Active research area with many extensions and related works:

Math foundation models

- MINERVA [Lewkowycz et al 2022] (*inaccessible*)
- LLEMMA [Azerbayev et al 2023]
- INTERNLM-MATH [Ying et al 2024]
- DEEPSEEK-MATH [Shao et al 2024]

Informal-to-formal translation and guidance

- Statements [Wu et al 2022]
- Verifying informal (DTV) [Zhou et al 2024]
- LLM Agent (COPRA) [Thakur et al 2024]
- LegoProver [Wang et al 2024]

Tools/case studies

- Codex autoformalization [Agrawal 2022]

Informal+formal benchmarks

- MINIF2F+INFORMAL [Jiang et al 2023]
- PROOFNET [Azerbayev et al 2023]
- MUSTARD [Huang et al 2024]

Data

- NATURALPROOFS-GEN [Welleck et al 2022]
- MMA [Jiang et al 2024]
- OpenWebMath [Paster et al 2023]
- Proofpile-II [Azerbayev et al 2023]

Other tutorials

- Neural theorem proving, IJCAI 2023
github.com/wellecks/ntptutorial
- ML for Theorem Proving, Neurips 2023
machine-learning-for-theorem-proving.github.io

Non-exhaustive list!

1. Intro: Foundation models \cap mathematics
 - Informal and formal mathematics
 - Why is formal mathematics important?
2. Part I: Build a LLM formal theorem proving tool
 - Data, training, proof search, evaluation, tool
3. Part II: Leveraging *informal* mathematical data
 - Via foundation model
 - Via translation and guidance

Thank you!

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

- Zhangir Azerbayev (Princeton)
- Stella Biderman (Eleuther)
- Jia Deng (Princeton)
- Marco Dos Santos (Cambridge)
- Jiewen Hu (CMU)
- Mateja Jamnik (Cambridge)
- Albert Jiang (Cambridge, Mistral)
- Timothee Lacroix (Mistral)
- Guillaume Lample (Mistral)
- Wenda Li (Edinburgh)
- Jiacheng Liu (Washington)
- Stephen McAleer (CMU)
- Keiran Paster (Toronto)
- Rahul Saha (Independent)
- Hailey Schoelkopf (Eleuther)
- Yuhuai (Tony) Wu (X.ai)
- Jin Zhou (Cornell)

<https://github.com/cmu-l3/ntptutorial-II>

<https://huggingface.co/l3lab>

Learning, Language, and Logic (L3) Lab



S. Welleck and R. Saha.

Llmstep: Llm proofstep suggestions in lean.

ArXiv, abs/2310.18457, 2023.



K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar.

LeanDojo: Theorem proving with retrieval-augmented language models.

In Neural Information Processing Systems (NeurIPS), 2023.



K. Zheng, J. M. Han, and S. Polu.

minif2f: a cross-system benchmark for formal olympiad-level mathematics.

In International Conference on Learning Representations, 2022.