LLMs and [in]formal mathematics

Daniel Fried, with slides from Sean Welleck

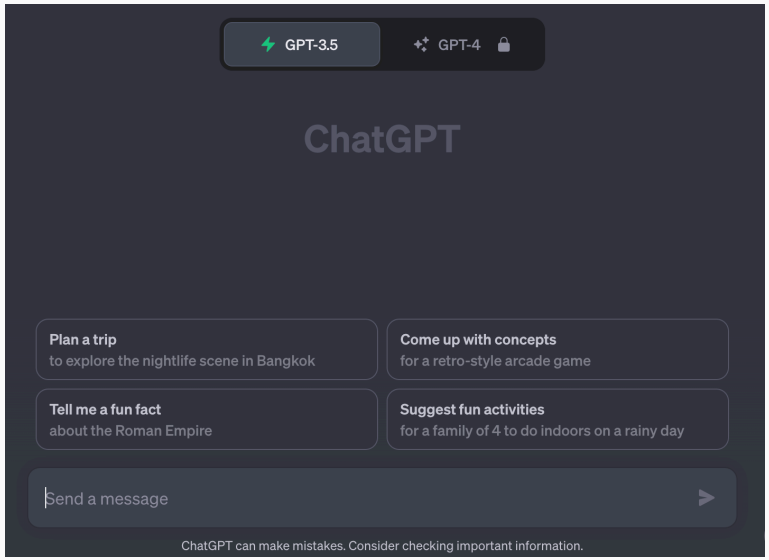
Nov 25, 2025

Carnegie Mellon University

Foundation models and large language models (LLMs)

Platform for sequence generation

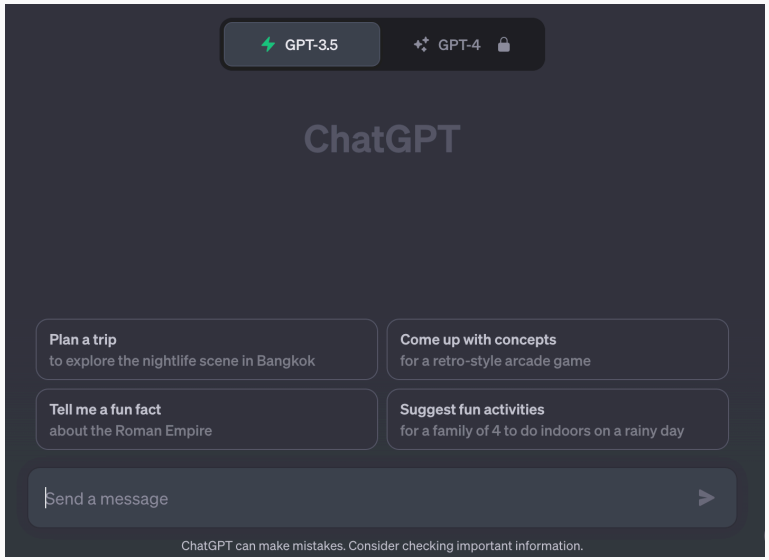
- Summarize documents
- Generate code from a screenshot
- ...



Foundation models and large language models (LLMs)

Core idea:

- Collect diverse data
- Train generative model
- Adapt to tasks (e.g., prompting)



Foundation models in expert domains

LLMs in expert domains

- Finance
- Medicine
- Mathematics

[1] See, 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), ...;

Foundation models in expert domains

LLMs in expert domains

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

[1] See, 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.

Formal mathematics

Math as **source code**

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

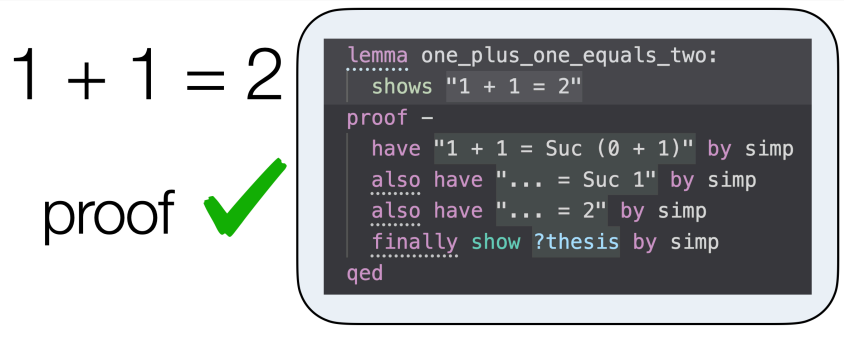


Figure 2: Math as source code.

Formal mathematics

Math as **source code**

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

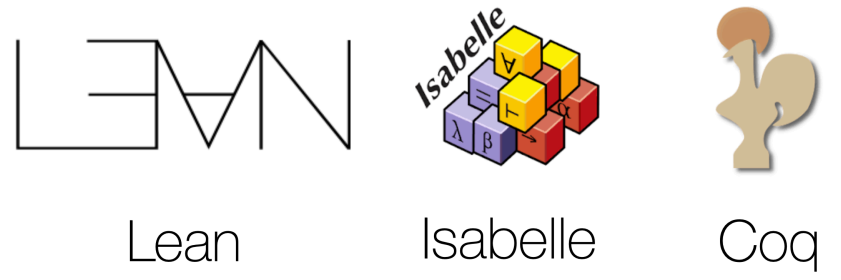
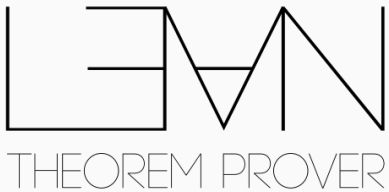


Figure 3: Theorem proving languages

Formal mathematics

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



How is formal math used in practice?

Growing use in mathematics:

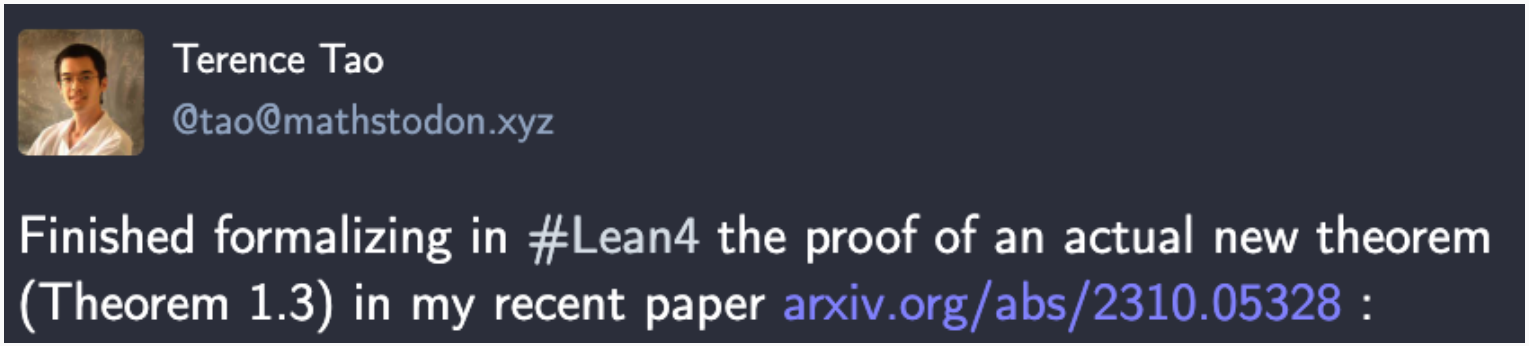


Figure 4: Terence Tao's Lean formalization project (October 2023)

How is formal math used in practice?

Growing use in mathematics:

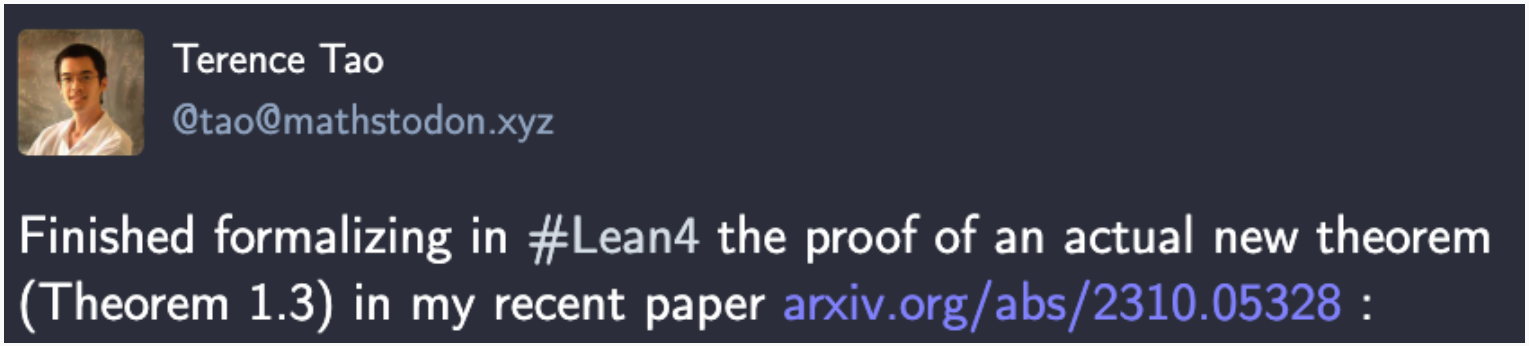


Figure 4: Terence Tao's Lean formalization project (October 2023)

- **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
- ...

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

Why is AI ∩ formal math important?

LLMs for formal math

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

Why is AI ∩ formal math important?

Formal math for LLMs

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

Why is AI ∩ 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 (*https://github.com/ImperialCollegeLondon/FLT*)

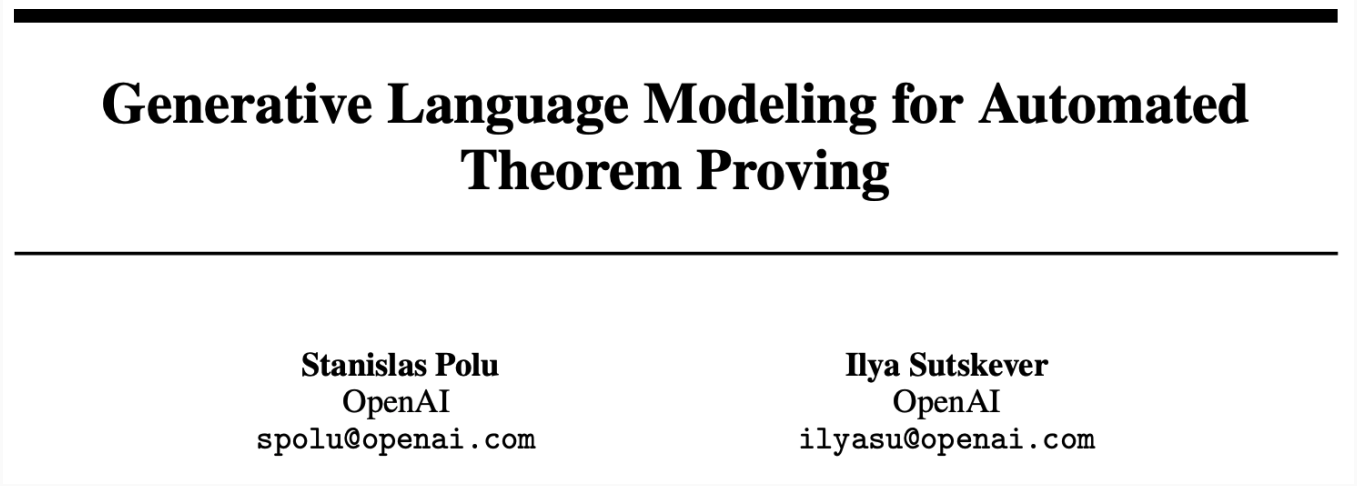
Why is AI ∩ formal math important?

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 (*https://github.com/ImperialCollegeLondon/FLT*)
- Complementary **tools**
  - Computer algebra, SAT/SMT solvers, ...
  - Rule-based automation, ...

And still far from being "solved" (April 2024)...

LLMs ∩ formal math



Generative Language Modeling for Automated Theorem Proving

Stanislas Polu
OpenAI
spolu@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Figure 5: *gpt-f* (2020)

LLMs ∩ formal math

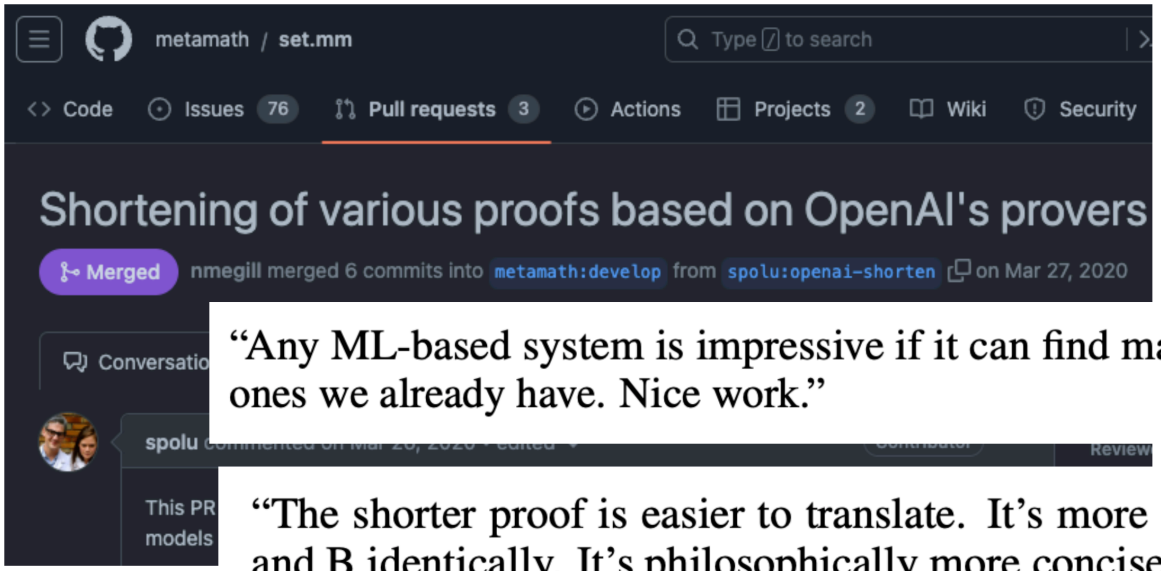


Figure 6: *gpt-f* (2020)

LLMs ∩ formal math

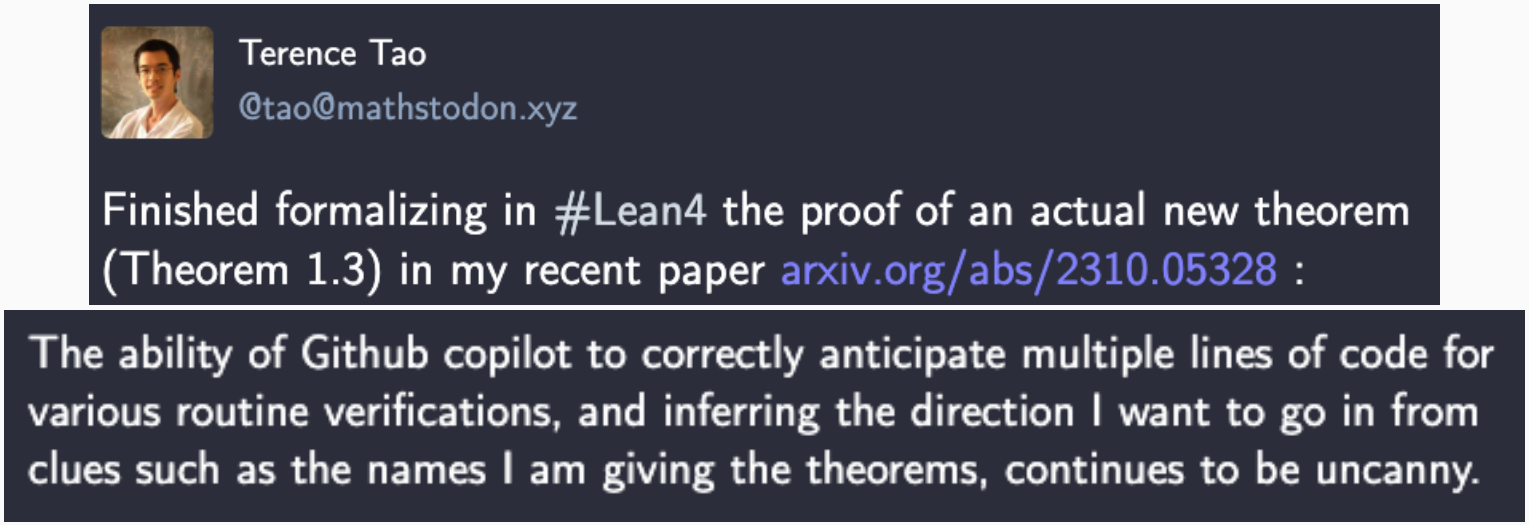


Figure 7: Terence Tao's Lean formalization project (October 2023)

This talk

1. **Intro: Foundation models ∩ mathematics**
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. Part I: Step-level theorem proving
   - Data, training, proof search, evaluation, tool

3. Part II: Formalization
   - Via translation and guidance

4. Part III: Whole-proof generation
   - Goedel Prover, Seed Prover

This talk

1. Intro: Foundation models ∩ mathematics
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. **Part I: Step-level theorem proving**
   - Data, training, proof search, evaluation, tool

3. Part II: Formalization
   - Via translation and guidance

4. Part III: Whole-proof generation
   - Goedel Prover, Seed Prover

This talk

1. Intro: Foundation models ∩ mathematics
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. Part I: Step-level theorem proving
   - Data, training, proof search, evaluation, tool

3. **Part II: Formalization**
   - Via translation and guidance

4. Part III: Whole-proof generation
   - Goedel Prover, Seed Prover

This talk

1. Intro: Foundation models ∩ mathematics
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. Part I: Step-level theorem proving
   - Data, training, proof search, evaluation, tool

3. Part II: Formalization
   - Via translation and guidance

4. **Part III: Whole-proof generation**
   - Goedel Prover, Seed Prover

Two approaches to LLM theorem proving

Step-level approaches

- Predict next tactic/step
- Use tree search (e.g., best-first)
- Iterative: state $\rightarrow$ step $\rightarrow$ new state
- Examples: Tutorial, ReProver, LLEMMA

Whole-proof generation

- Generate complete proof end-to-end
- Can use verifier feedback for correction
- Compatible with chain-of-thought reasoning
- Examples: Goedel Prover, Seed Prover

Part I:
Step-level theorem proving

Build a [L]LM proving tool[2]

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

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

Build a [L]LM proving tool[3]

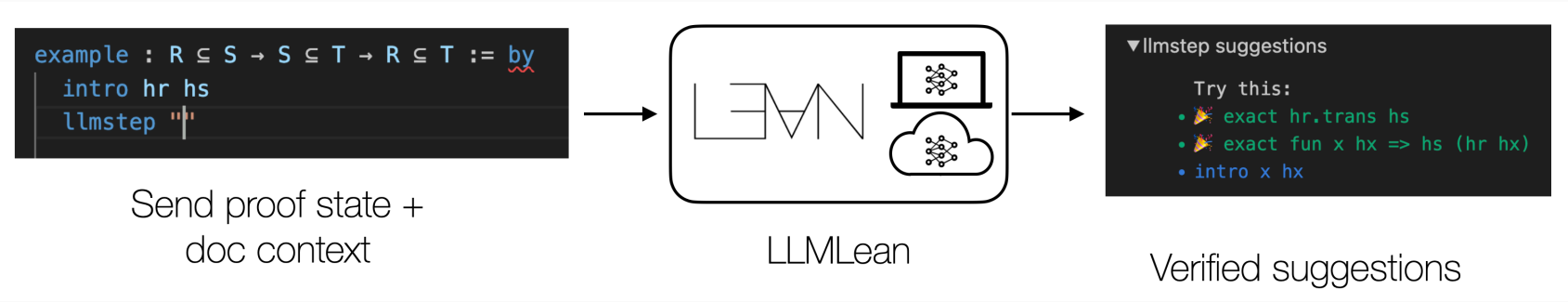
Artifacts:

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

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

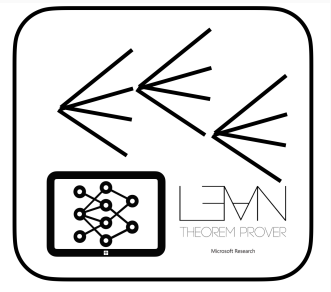
Build a [L]LM proving tool[4]



[4] Update 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



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

Language models

- 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

Language models

- 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:
  - $\arg\max_\theta \sum_{\mathbf{y}\in\mathcal{D}} \log p_\theta(\mathbf{y})$

Language models

- 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:
  - $\arg\max_\theta \sum_{\mathbf{y}\in\mathcal{D}} \log p_\theta(\mathbf{y})$

Language models

- 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:
  - $\mathbf{y} = f(p_\theta(\cdot|x))$
  - f: e.g., sampling

Problem setup

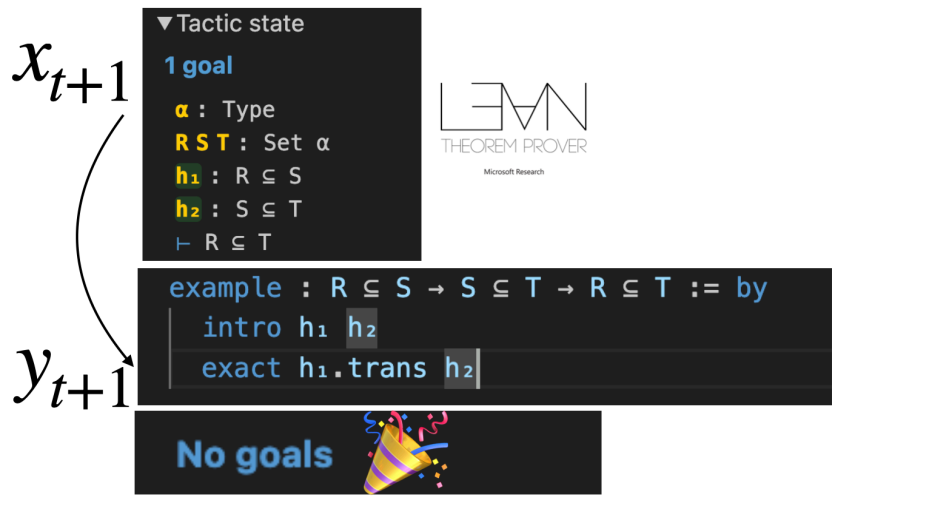
Proof: sequence of (state, step)

- $(x_0, y_0), \ldots, (x_T, y_T)$
- x_t: proof state from Lean
- y_t: proof step ("tactic")

Problem setup

Proof: sequence of (state, step)

- $(x_0, y_0), \ldots, (x_T, y_T)$
- x_t: proof state from Lean
- y_t: proof step ("tactic")



Problem setup

Proof: sequence of (state, step)

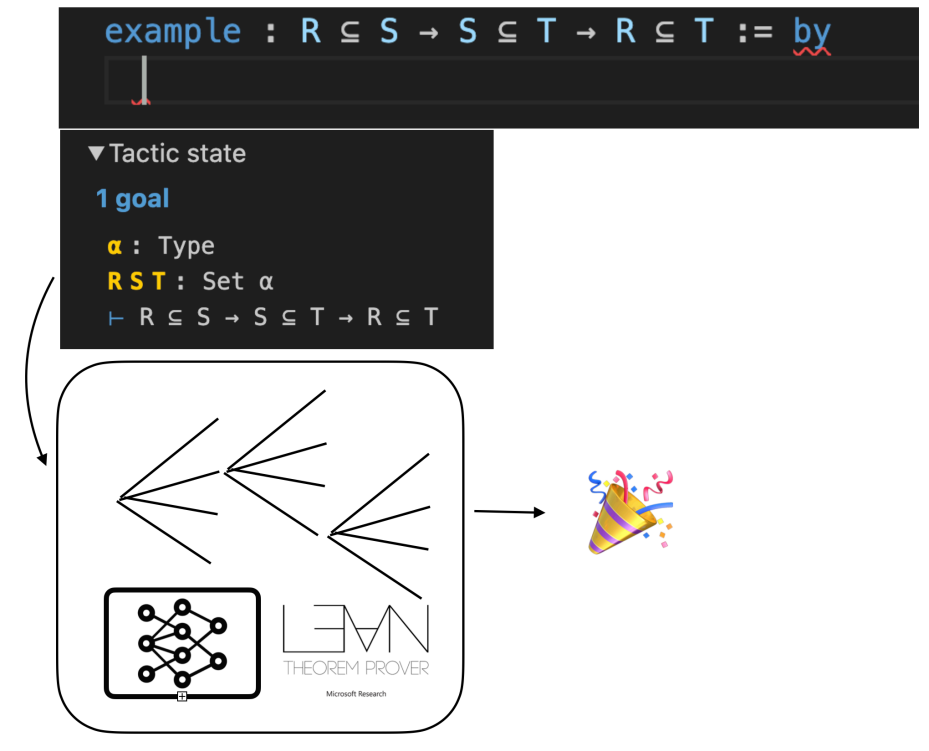
- $(x_0, y_0), \ldots, (x_T, y_T)$
- x_t: proof state from Lean
- y_t: proof step ("tactic")

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

Problem setup

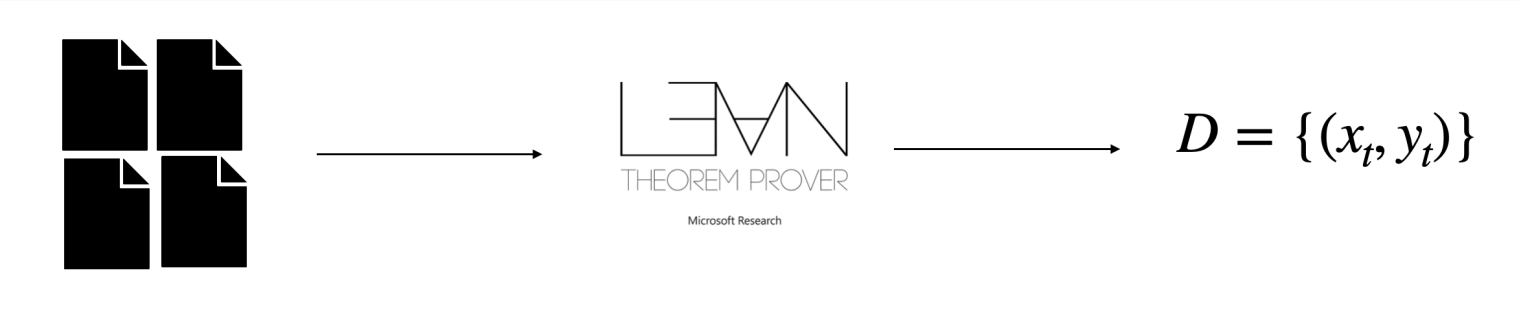
Proof: sequence of (state, step)

- $(x_0, y_0), \ldots, (x_T, y_T)$
- 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]

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

1. Data (ntp-training-data)

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

Prompt:

```
/- 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]
```

Completion:

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

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

1. Data (ntp-training-data)

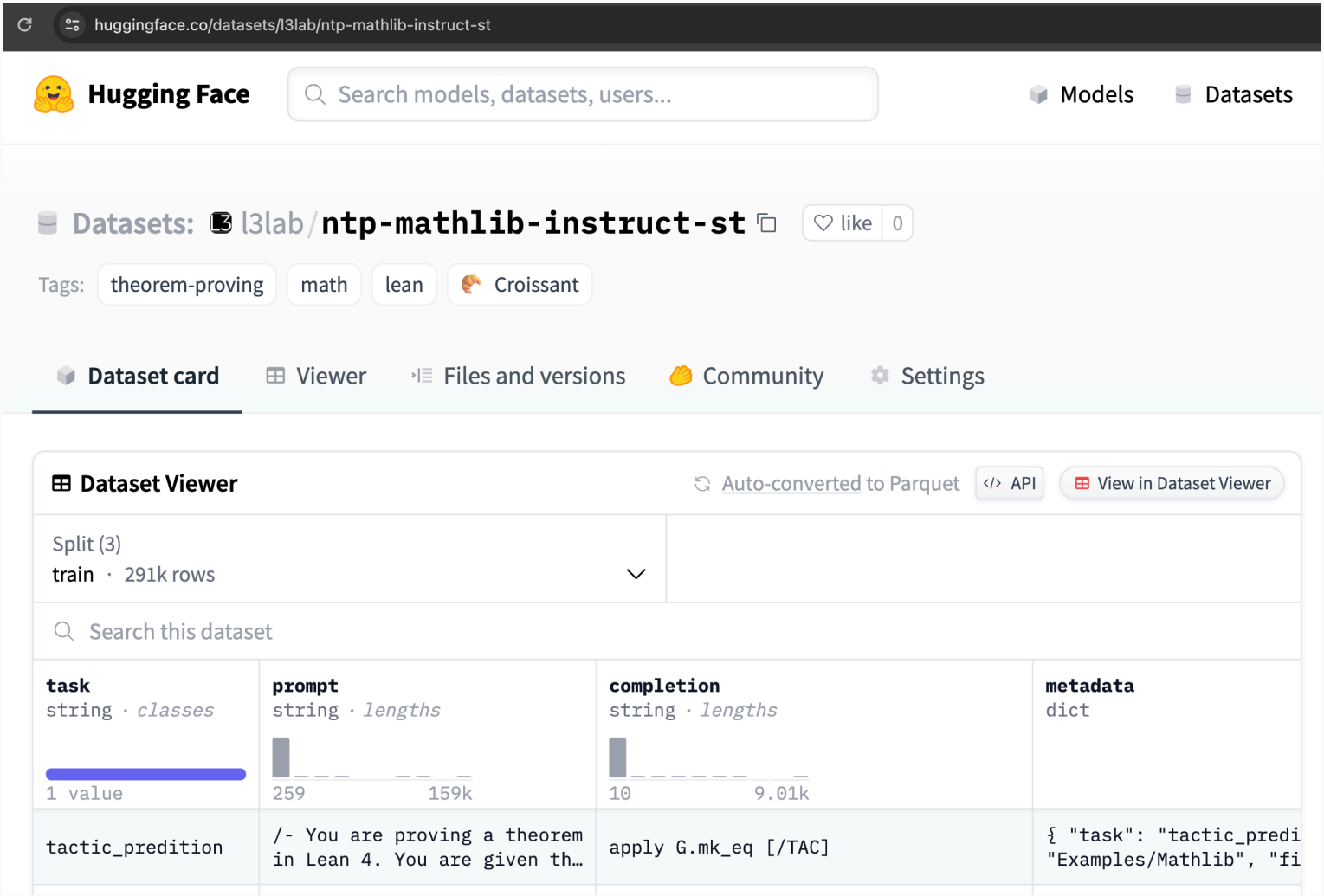


Figure 8: Data on Huggingface

[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)

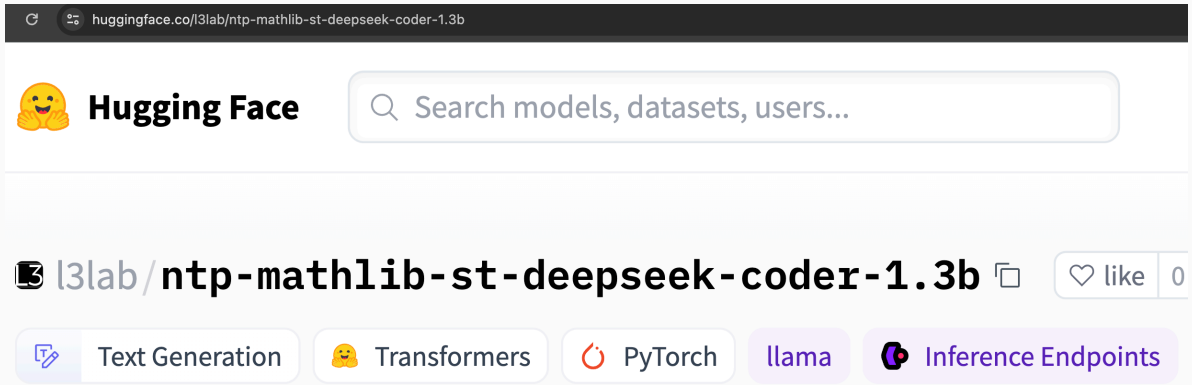
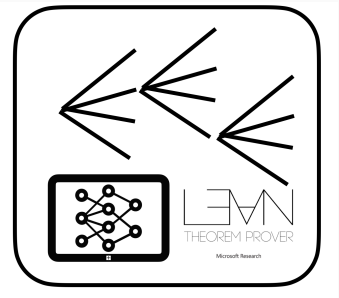


Figure 9: Trained tutorial model on Huggingface

[NOTEBOOK DEMO]

3. Proof search

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



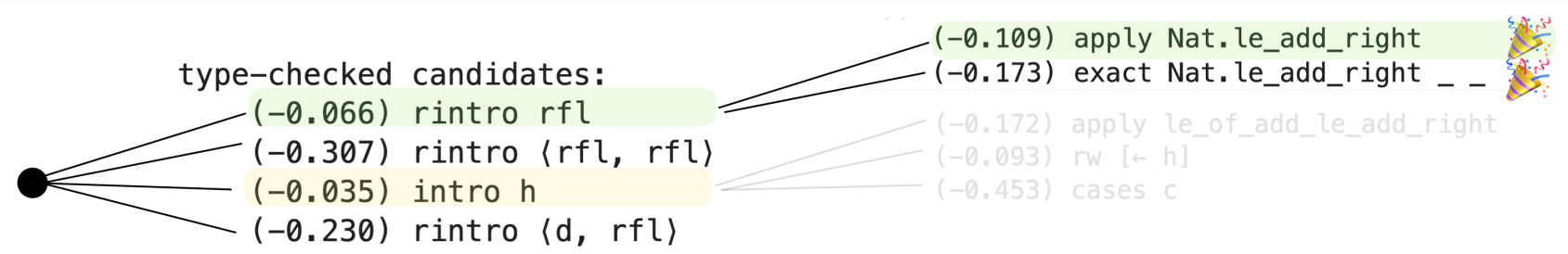


Figure 10: Best-first search[6]

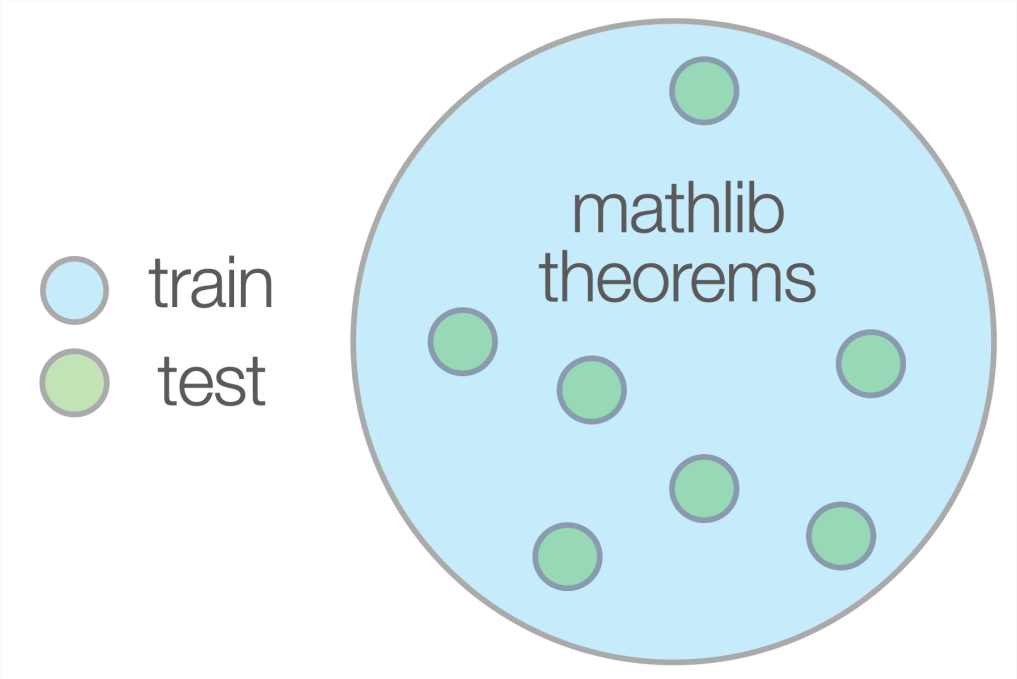
[6] Example scoring function $\frac{1}{Z}\sum_t \log p_\theta(y_t|x_t)$

3. Proof search | Best-first search

[NOTEBOOK DEMO]

4. Evaluation

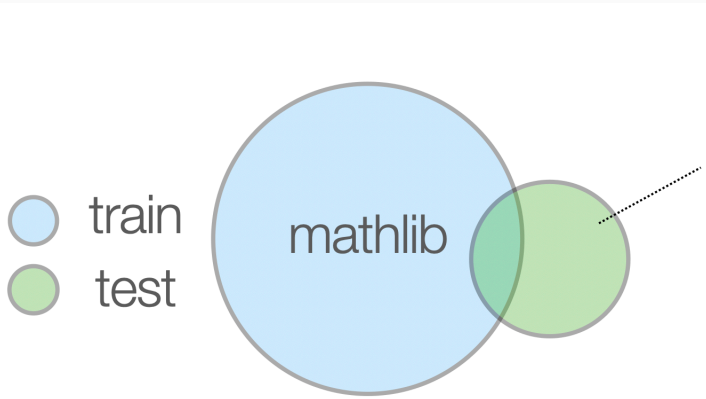
- Proof search on held-out theorems from the training distribution



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
```

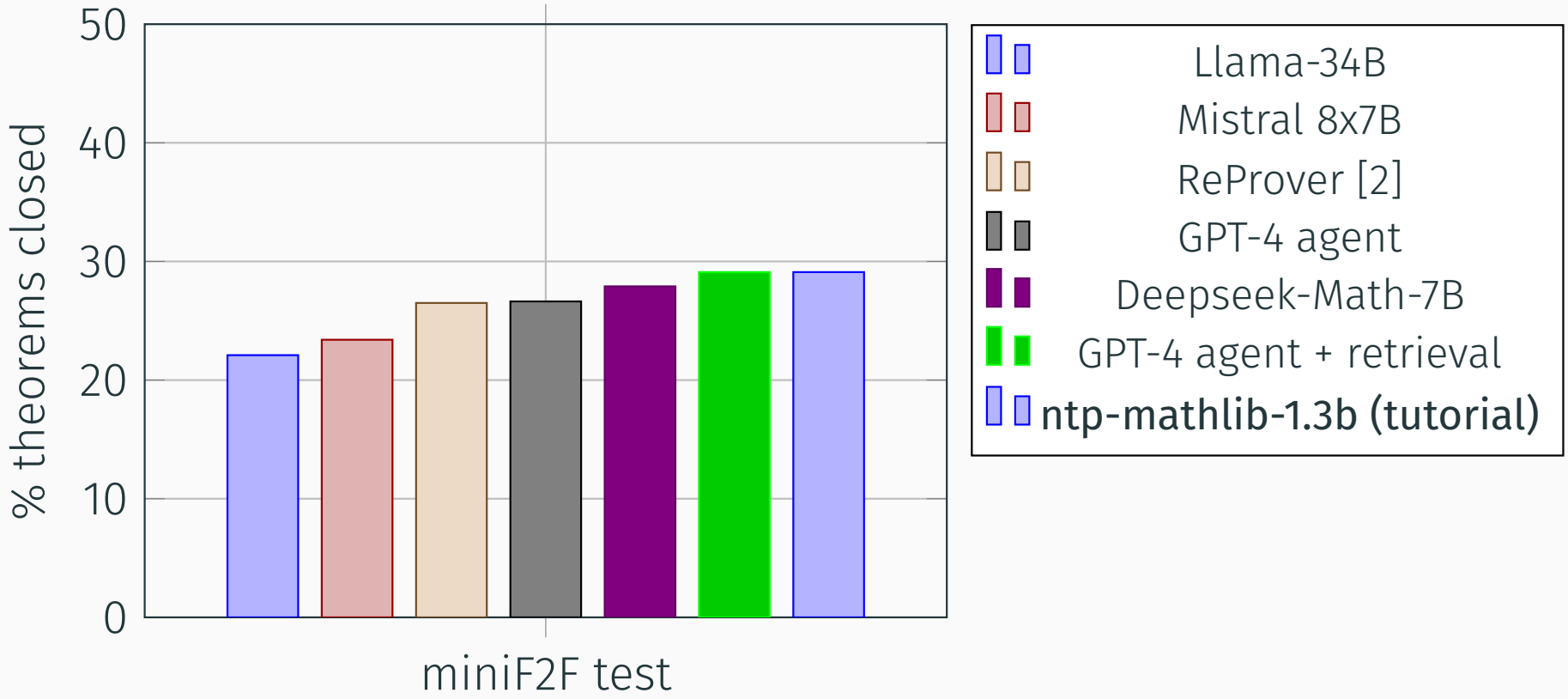


Figure 11: Proof search performance on miniF2F theorems. **The model we trained in the tutorial notebooks gets 29.1% (71/244) on miniF2F test.**

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

4. Evaluation

[NOTEBOOK DEMO]

5. Extensions: context

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

```
variable {Ω : Type*}[Fintype Ω]

structure my_object (Ω : Type*)[Fintype Ω] :=
  (f : Ω → ℝ)
  (cool_property : ∀ x : Ω, 0 ≤ f x)

theorem my_object_sum_nonneg (o1 o2: my_object Ω) : o1.f + o2.f ≥ 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

Prompt:

```
/- 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]
```

Completion:

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

[NOTEBOOK DEMO]

6. Integrating LLMs and Lean

LLMLEAN: tools that integrate LLMs into the Lean proof assistant

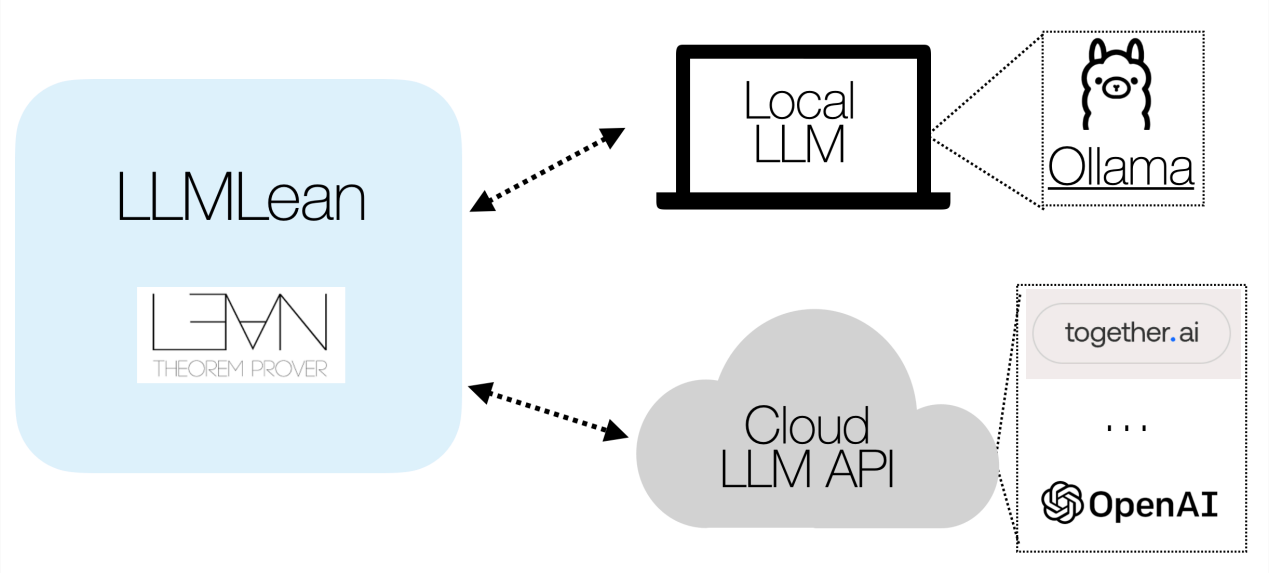


Figure 14: github.com/cmu-l3/llmlean

6. Integrating LLMs and Lean

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

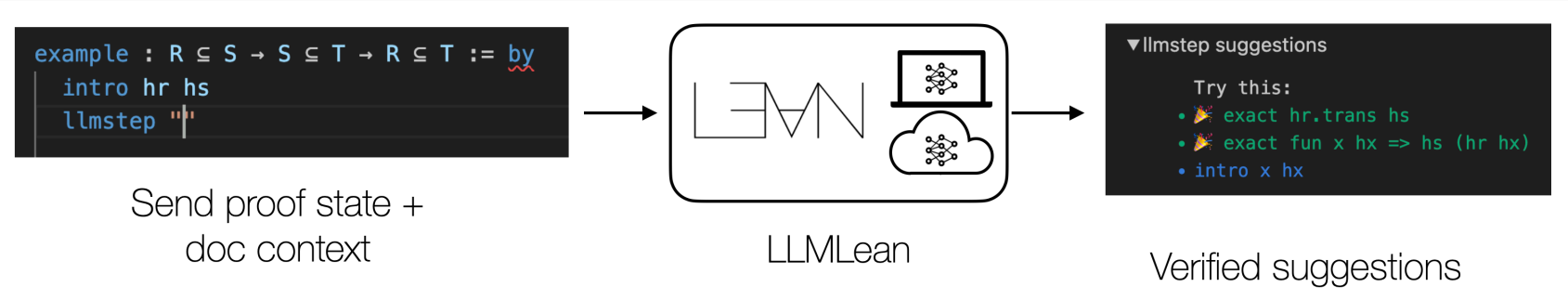


Figure 15: github.com/cmu-l3/llmlean

[DEMO]

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

Recap

- 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

Recap

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

- Reprover [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 [Azerbayev 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!

This talk

1. Intro: Foundation models ∩ mathematics
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. Part I: Step-level theorem proving
   - Data, training, proof search, evaluation, tool

3. **Part II: Formalization**
   - Via translation and guidance

4. Part III: Whole-proof generation
   - Goedel Prover, Seed Prover

Part II: Formalization

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, ...

Leveraging *informal* mathematical data

Why leverage informal data?

1. Data scarcity
    - Lean: 300 million tokens
    - Arxiv: 29 billion tokens
    - General web data: > 5 trillion tokens

Leveraging *informal* mathematical data

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

Leveraging *informal* mathematical data

Why leverage informal data?

1. Data scarcity
   - Lean: 300 million tokens
   - Arxiv: 29 billion tokens
   - General web data: > 5 trillion tokens

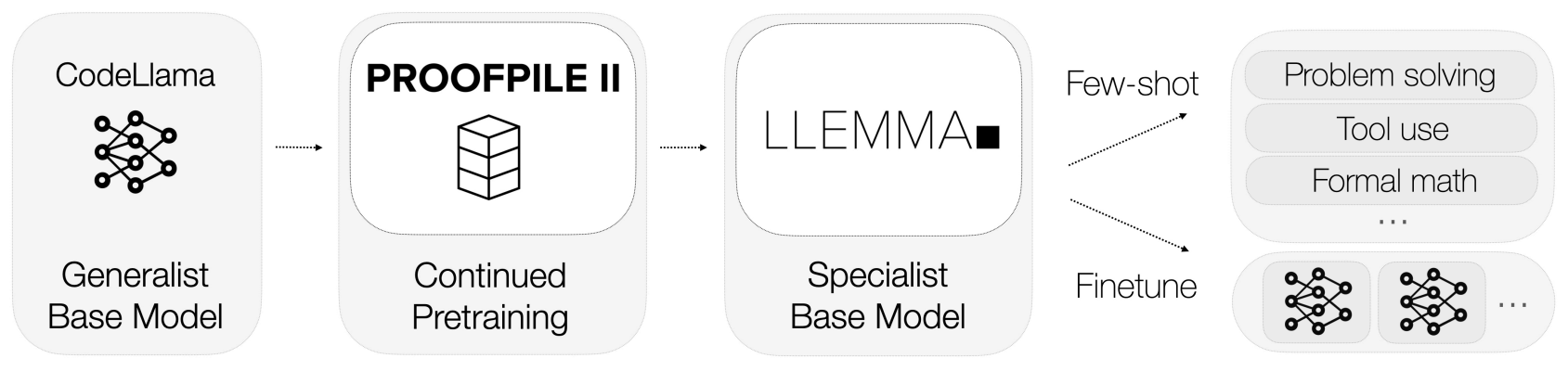
   $\rightarrow$ ***transfer knowledge by adapting a generalist model to mathematical data***

2. Guiding search
   - Informal proofs can help cut down the space of possible proofs

Foundation model for mathematics : LLEMMA[8]

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

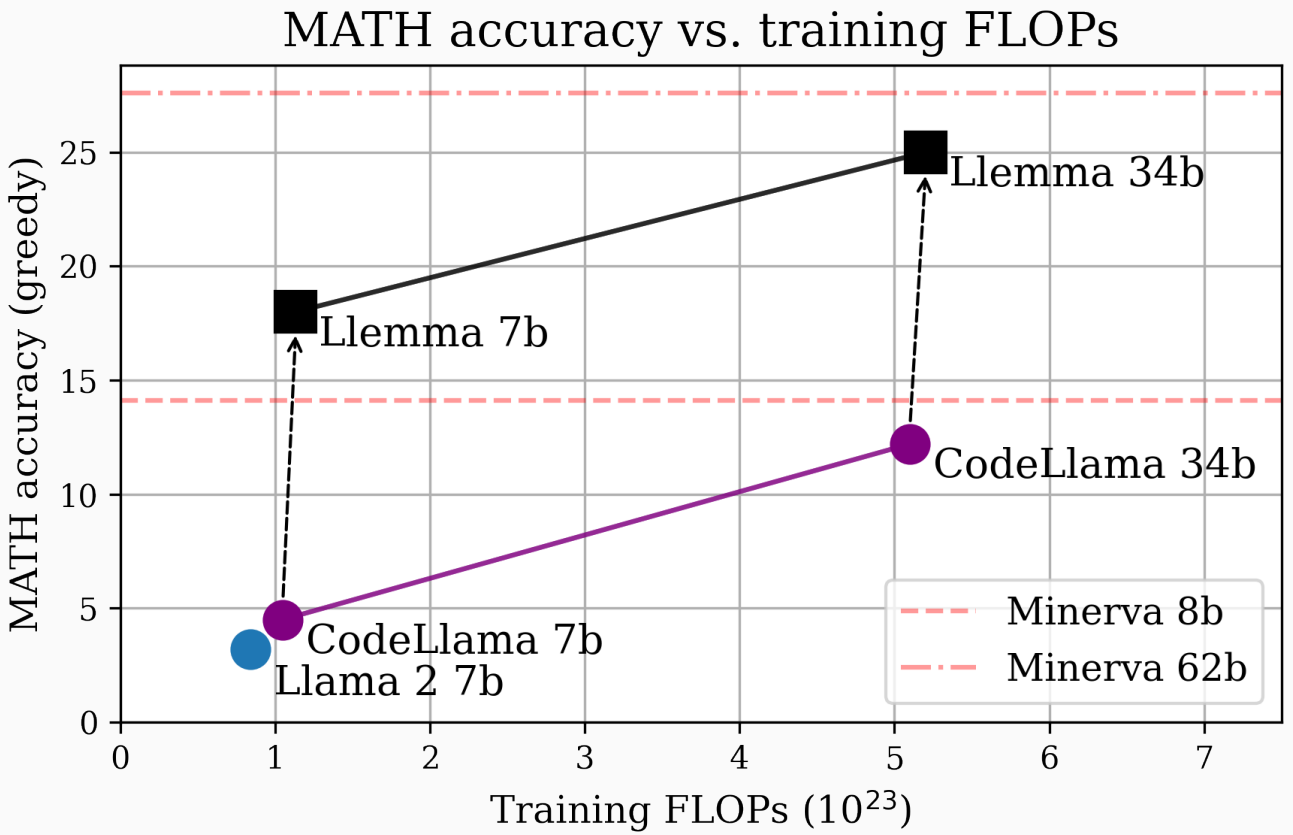


Figure 16: LLEMMA improves with a modest amount of math-specific compute

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

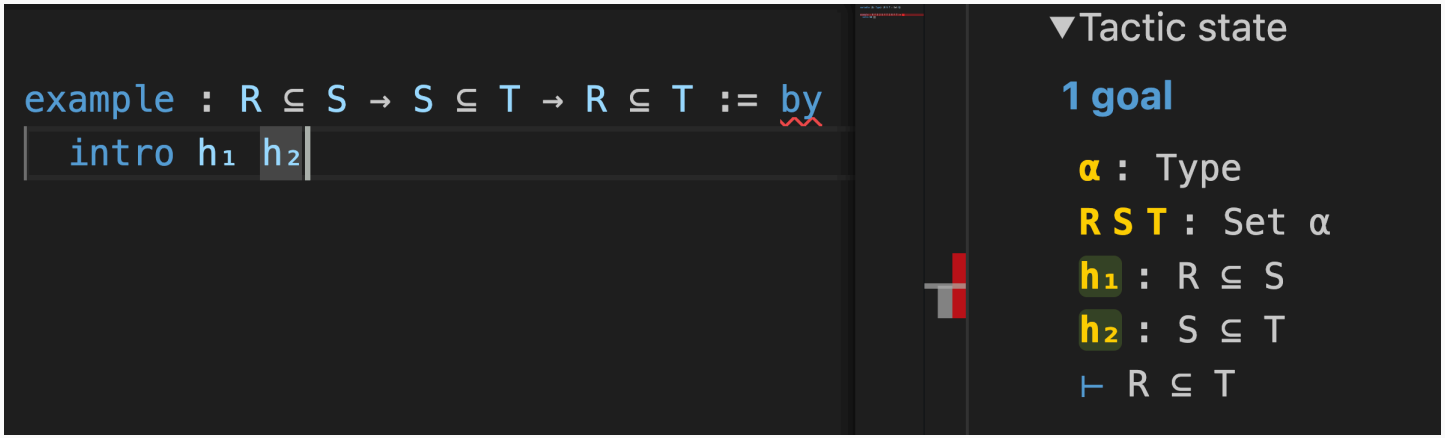


Figure 17: Lean code (left) and goal state (right)

Llemma | web: OpenWebMath [Paster et al 2023][9]

- 14.7 billion tokens of math-related web data

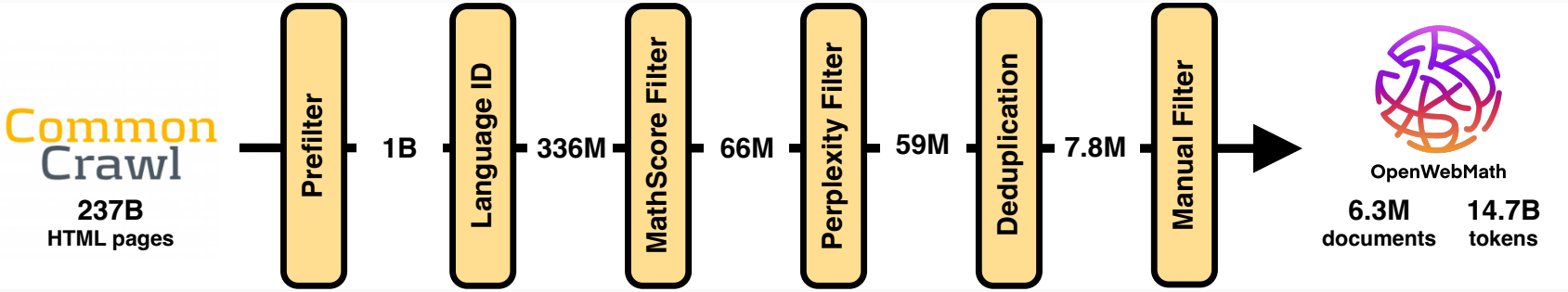


Figure 18: OpenWebMath pipeline.

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

Traditional proof search: p_θ(next-tactic|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

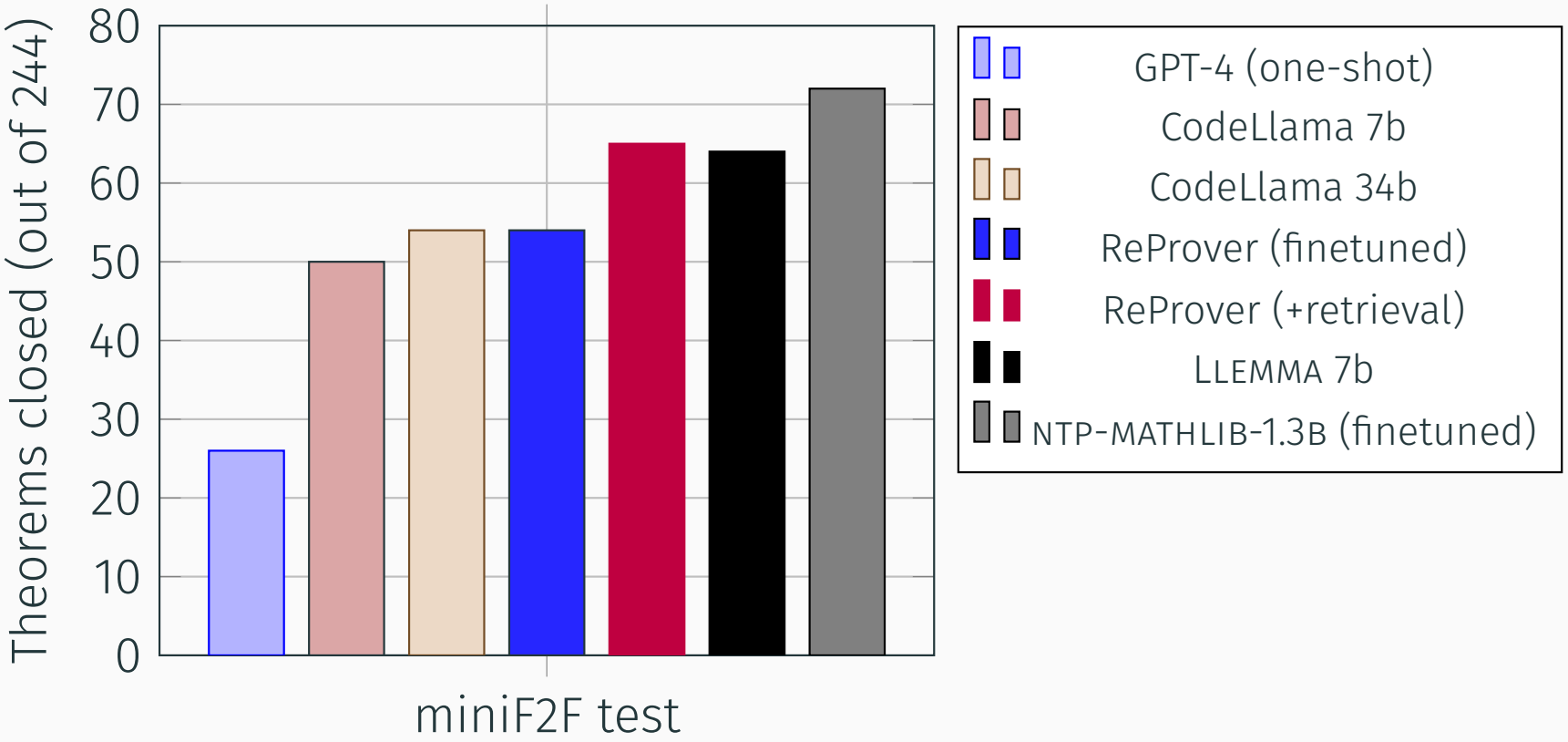
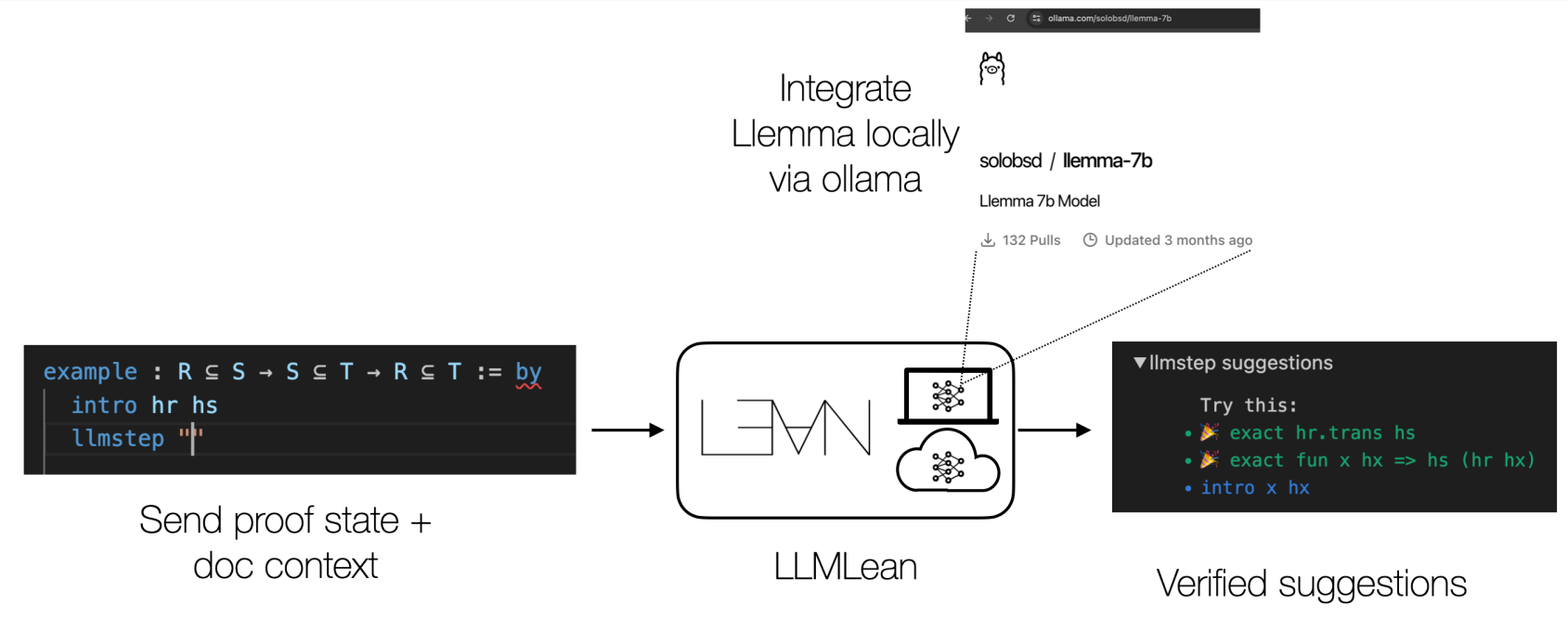


Figure 19: Few-shot proving in Lean with LLEMMA

LLEMMA on your laptop with LLMLEAN [10]



DEMO

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

Recap

- 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*

Leveraging *informal* mathematical data

Why leverage informal data?

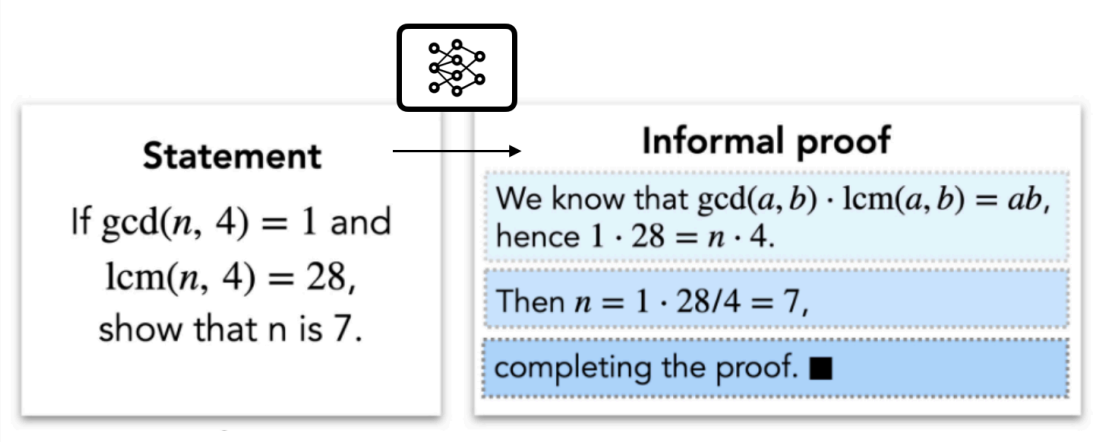
1. Data scarcity

2. **Guiding search**
   - **Informal proofs can help cut down the space of possible proofs**

Draft-sketch-prove[11]

Given informal theorem x_I,
formal theorem x_F

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

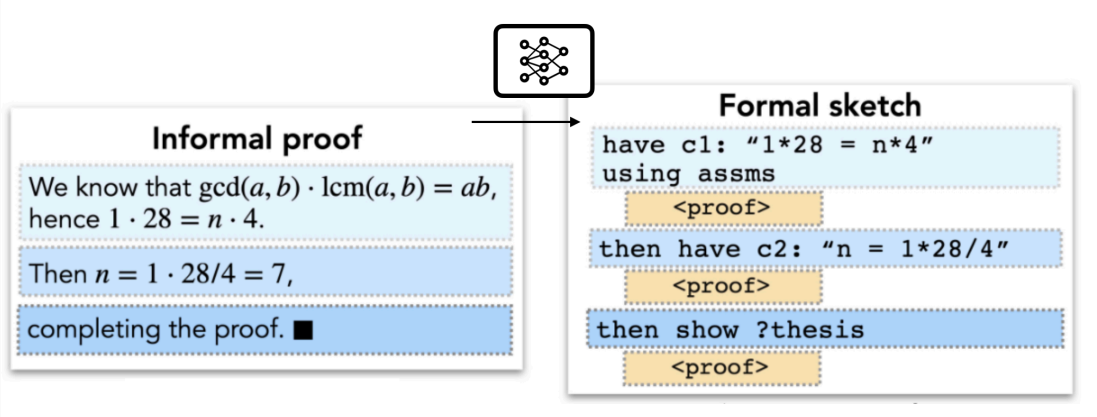


[11] *Draft, Sketch, Prove: Guiding Formal Theorem Provers with Informal Proofs*
Jiang, Welleck, Zhou, Lacroix, Liu, Li, Jamnik, Lample, Wu. ICLR 2023

Draft-sketch-prove[11]

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

Draft-sketch-prove[11]

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

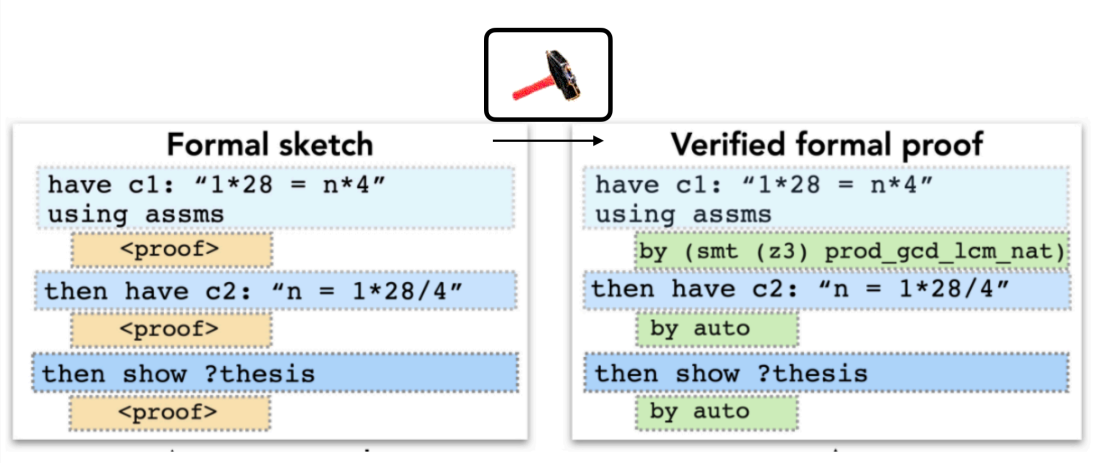


Figure 20: "Classical" prover *Sledgehammer*

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

Draft-sketch-prove

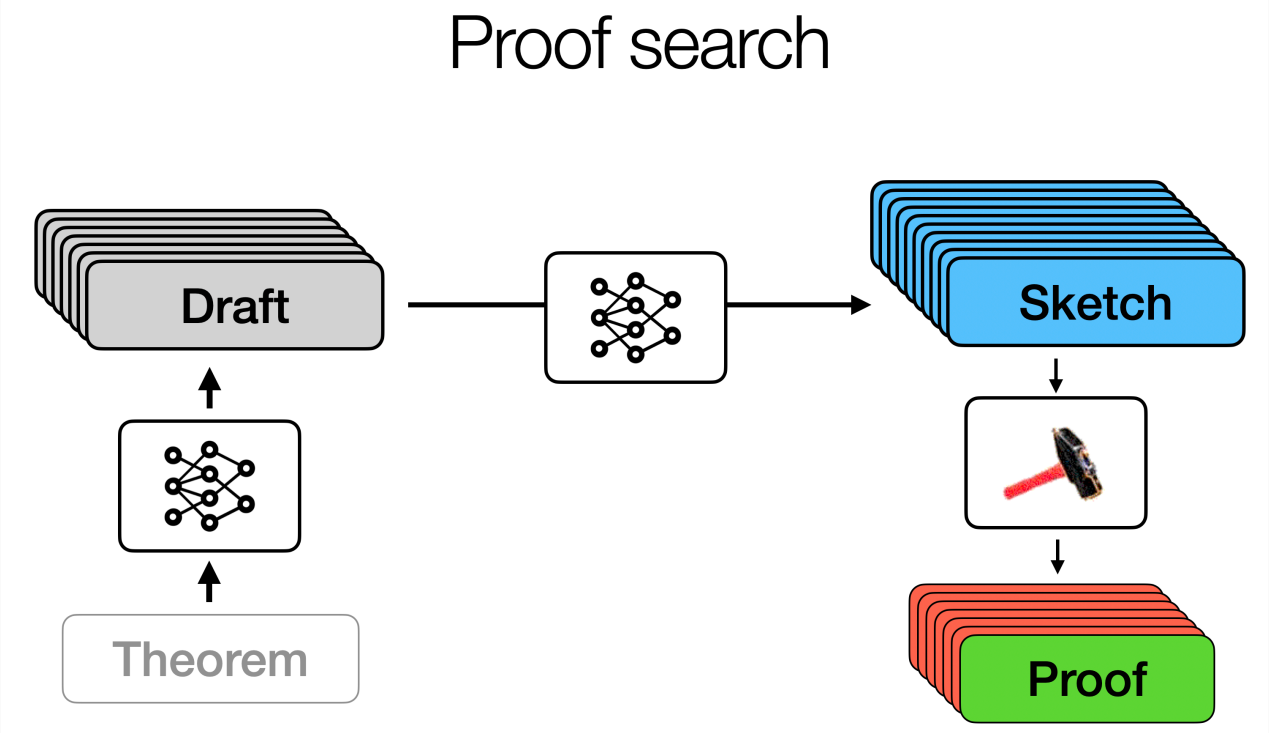


Figure 21: Proof search

Draft-sketch-prove

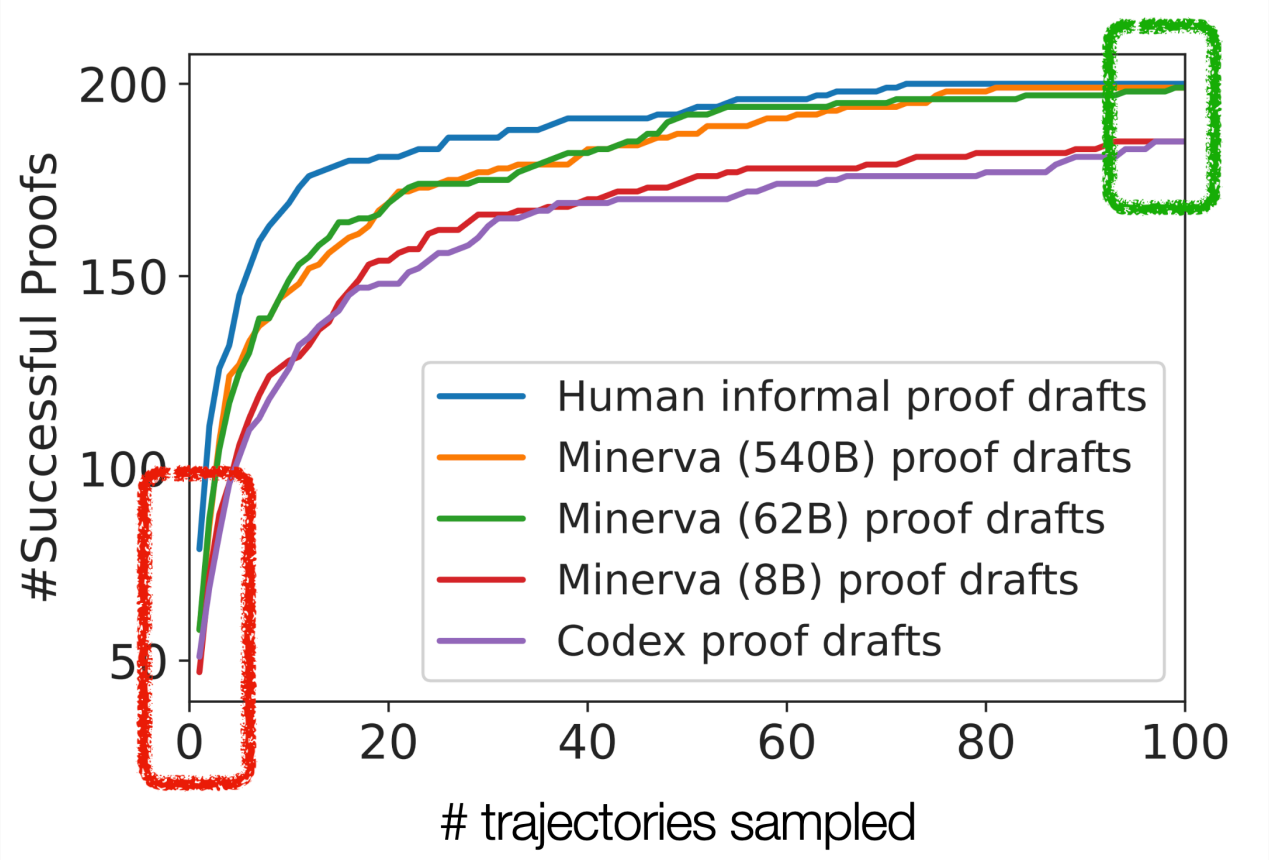
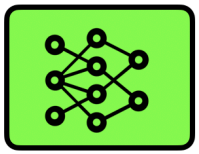
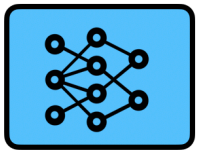


Figure 22: Proof search scaling

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
```

Figure 23: International Math Olympiad problem

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!

Part III: Whole-proof generation

Whole-proof generation

Approach: Generate complete proofs, rather than feeding intermediate steps to the interpreter

Whole-proof generation

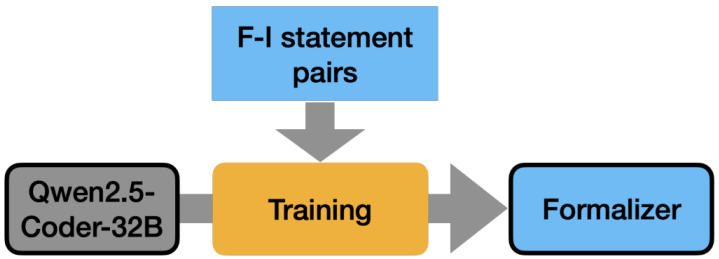
Approach: Generate complete proofs, rather than feeding intermediate steps to the interpreter

Advantages:

- Better fit for large-scale training with synthesized data / RL
- Can still apply interpreter feedback at the level of proofs
- Compatible with search

Goedel Prover: Training a formalizer

- Train a “formalizer” model to generate formal statements from informal. Use Lean workbook and Claude sonnet for training data
- Informal statements are abundant online. Numina dataset: hundreds of thousands of math competition etc problems.
- Use the trained model to formalize **1.64 million** statements
- Will use this to train a proof generation model.

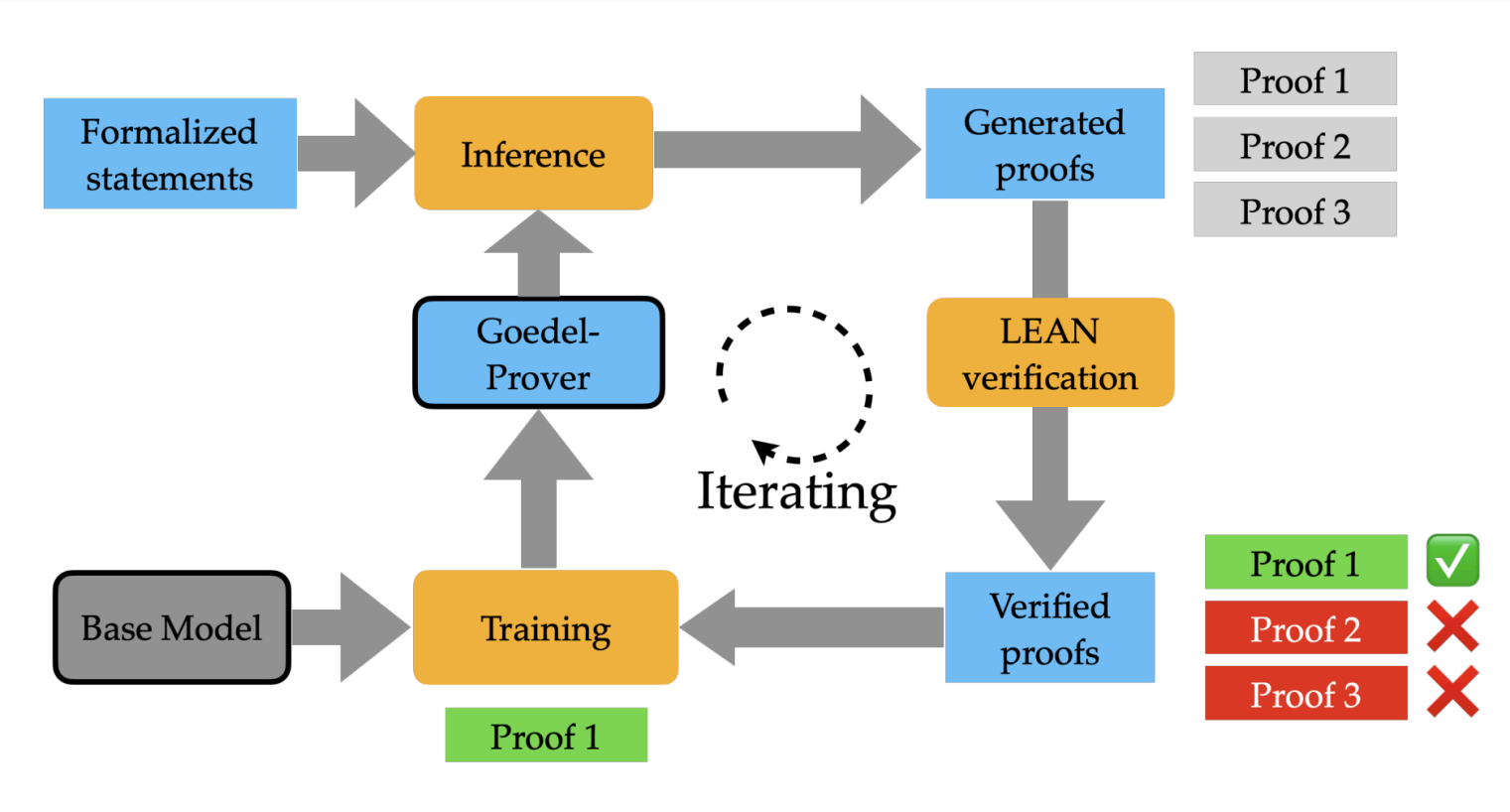


Formal and Informal (F-I) statement pair

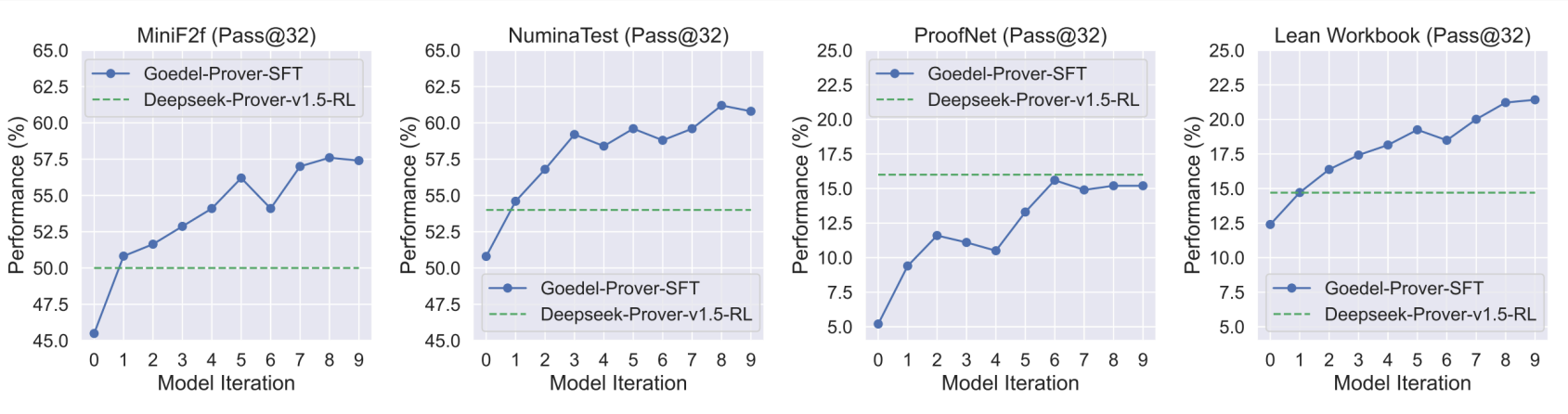
Informal statement	Let a, b, and c be real numbers that satisfying $3a+b+c=-3$, $a+3b+c=9$, and $a+b+3c=19$. Prove that the product $abc=-56$.
Formal statement	`theorem mathd_algebra_338 (a b c : ℝ) (h₀ : 3 * a + b + c = -3) (h₁ : a + 3 * b + c = 9) (h₂ : a + b + 3 * c = 19) : a * b * c = -56 := by`

Goedel Prover: Training a prover

1. Prover generates whole proofs for statements.
2. Lean compiler verifies correctness.
3. Correct proofs are added back to training data; model is fine-tuned.
4. Repeat.



Goedel Prover: Results



Seed Prover: Lemma-style proving

- Instead of generating a single theorem block, first generate useful intermediate lemma blocks.
- Modularity:
  - Enables clear identification of lemmas that are successfully proved versus those needing refinement.
  - Lemmas can be compiled independently, stored, and combined across problems.

Whole proof

```
theorem eg : 1 + 2 = 3 := by
  have h0 : 1 + 1 = 2 := by
    ring

  have h1 : 1 + (1 + 1) = 3 := by
    ring

  linarith [h0, h1]

#print axioms eg
```

Lemma-style

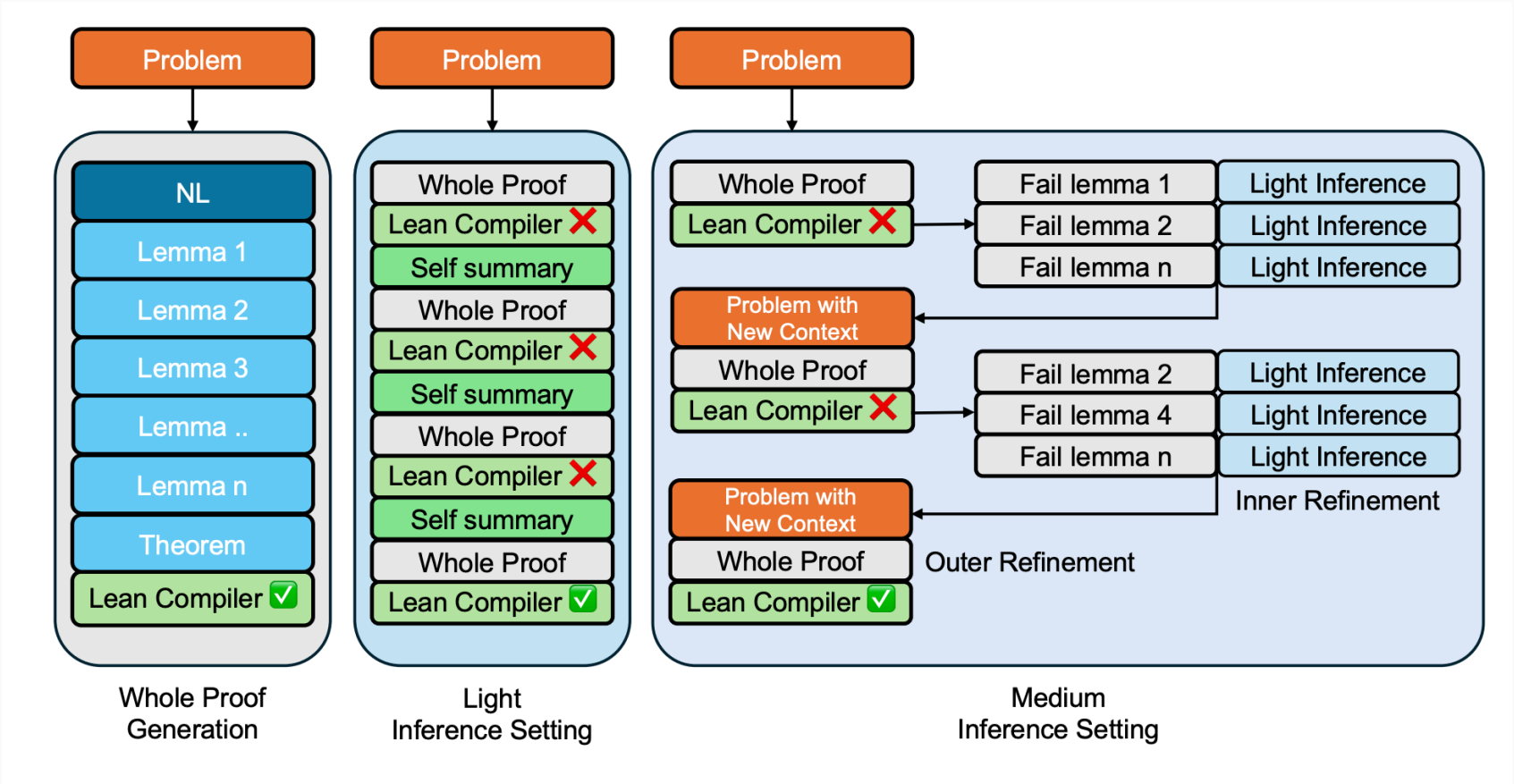
```
lemma round1_h0 : 1 + 1 = 2 := by
  ring

lemma round1_h1 : 1 + (1 + 1) = 3 := by
  ring

theorem eg : 1 + 2 = 3 := by
  linarith [round1_h0, round1_h1]

#print axioms eg
```

Seed Prover: Training

- Trains with RL (VAPO); uses binary rewards from the Lean compiler.
- No details about training data :(
- Chains of thought include conjectures in natural language
- Explicitly rewarded for using the `lemma` keyword before stating the main theorem.
- Prompt "*randomly incorporates natural language hints, natural language proofs, similar lemmas, proved lemmas, failed lemmas, failed attempts, summaries of previous attempts, and Lean compiler feedback into the prompt*"

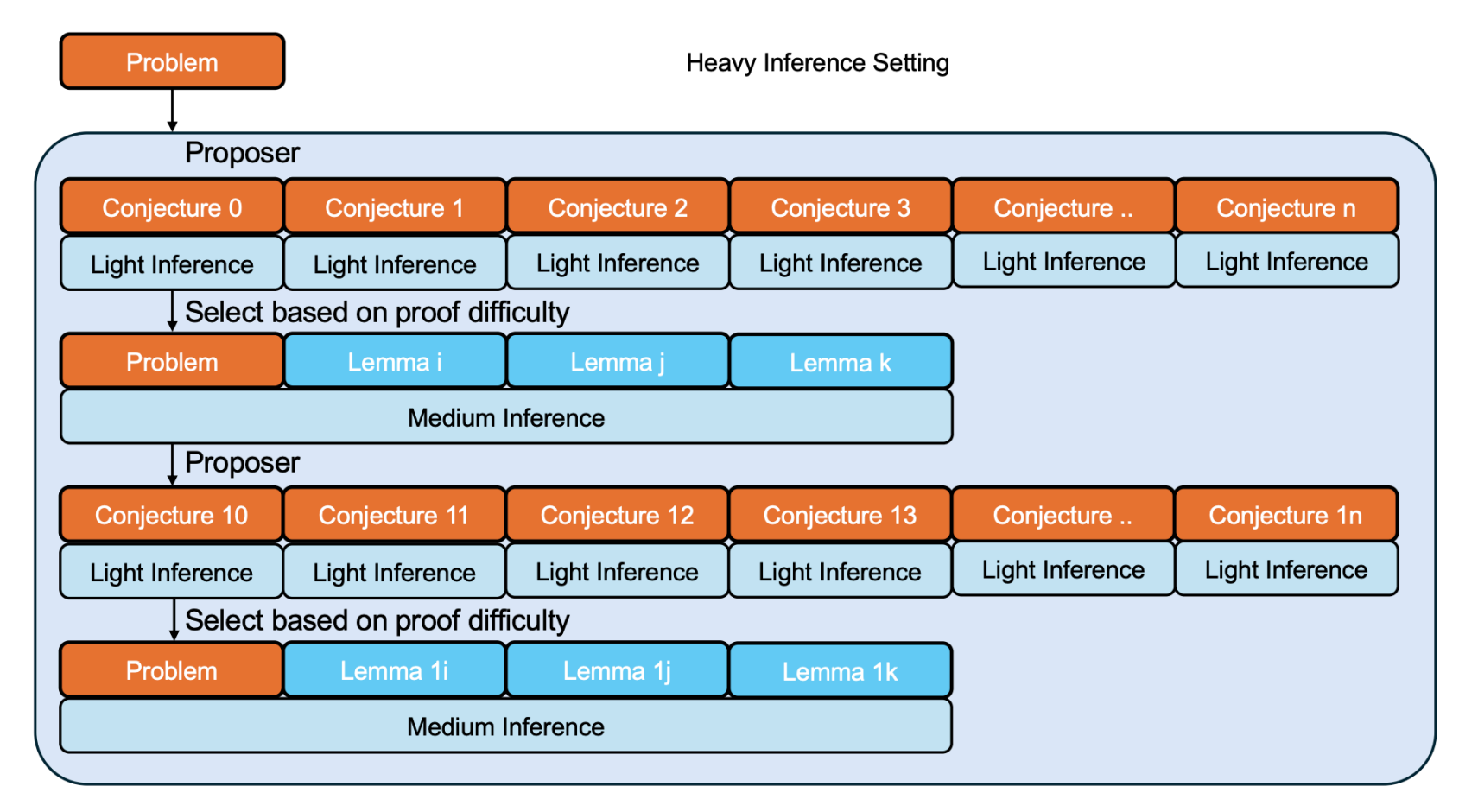
Seed Prover: Inference

- **Light:** Iterative refinement using compiler feedback (8-16 refinements x 8-16 generations). 1-2 hours.
- **Medium:** Recursive refinement for challenging intermediate lemmas.



Seed Prover: Inference

- **Heavy:** Also generates conjectures (5K) and uses a lemma pool. ‘Days of thinking’.



Seed Prover: Results

- Proved 5 out of 6 IMO 2025 problems and 78.1% of past IMO problems.
- Sample proof: *https://github.com/ByteDance-Seed/Seed-Prover/blob/main/SeedProver/imo2025/p1_proof.pdf*
- Achieved 99.6% success on MiniF2F-test, effectively saturating the standard benchmark.

Recap

1. Intro: Foundation models ∩ mathematics
   - Informal and formal mathematics
   - Why is formal mathematics important?

2. Part I: Step-level theorem proving
   - Data, training, proof search, evaluation, tool

3. Part II: Formalization
   - Via translation and guidance

4. Part III: Whole-proof generation
   - Goedel Prover, Seed Prover

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

Sean Welleck (CMU)
Learning, Language, and Logic (L3) Lab

References i


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.