LeanDojo: Theorem Proving with Retrieval-Augmented Language Models

Kaiyu Yang

Postdoc @ Computing + Mathematical Sciences

Caltech

Theorem Proving in Proof Assistants

Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
```

Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
```

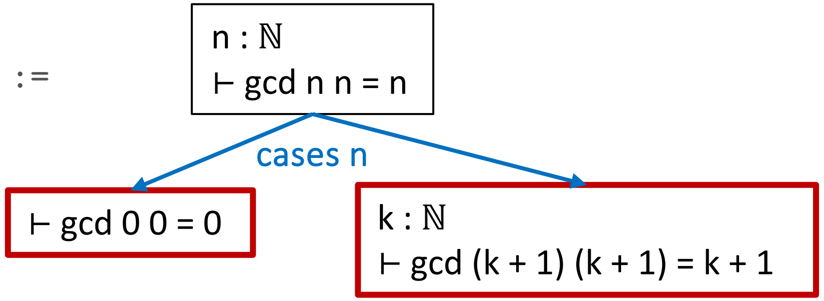
$n : \mathbb{N}$
$\vdash \text{gcd } n\ n = n$

Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
```

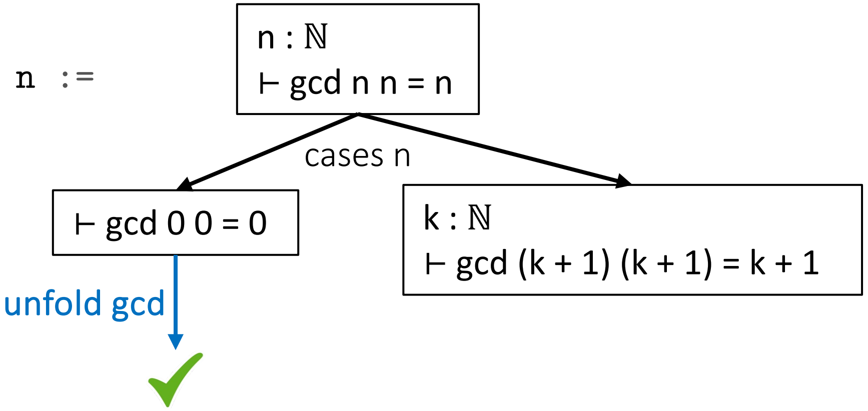


Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
```

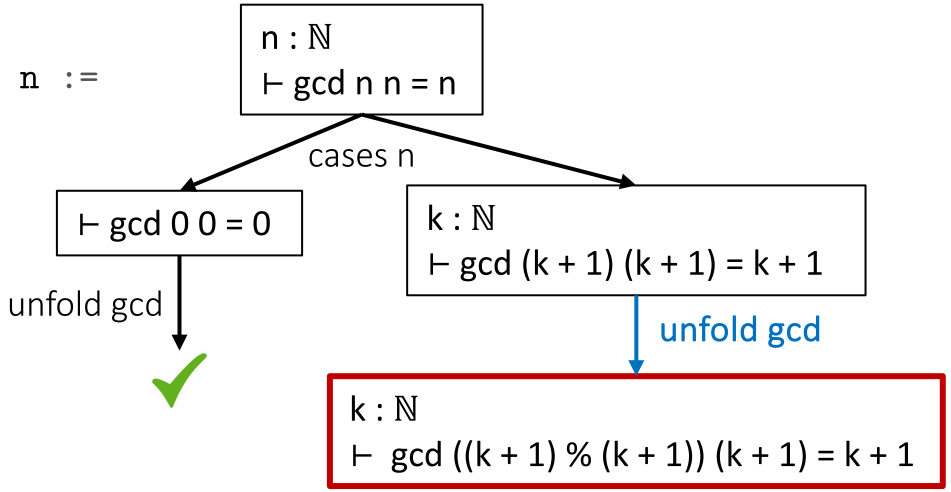


Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
```

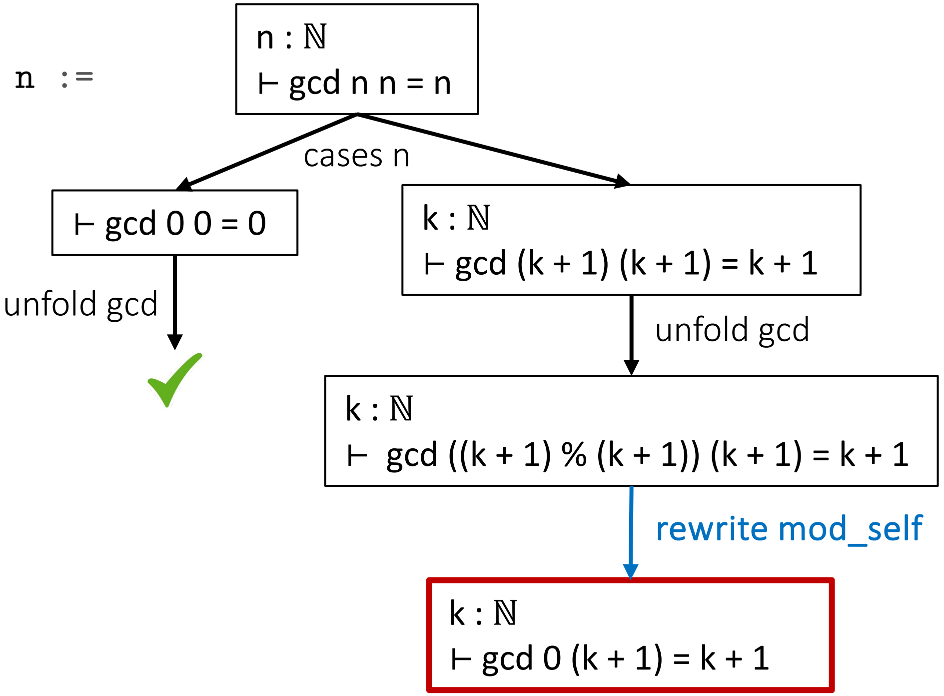


Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
```

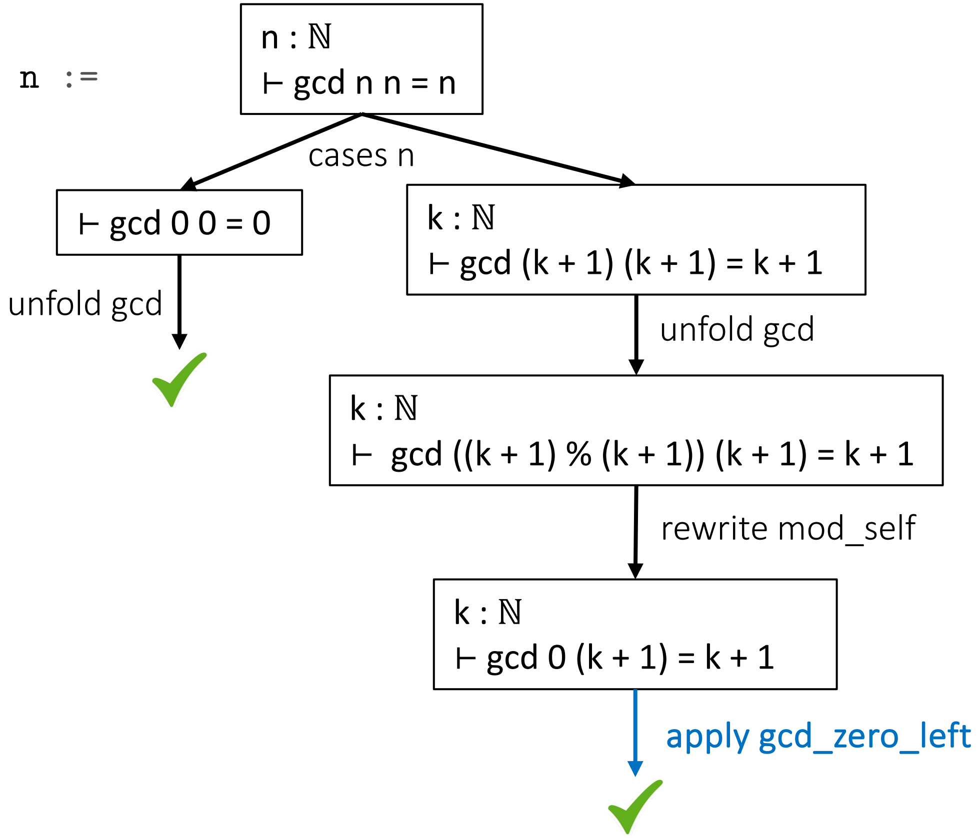


Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

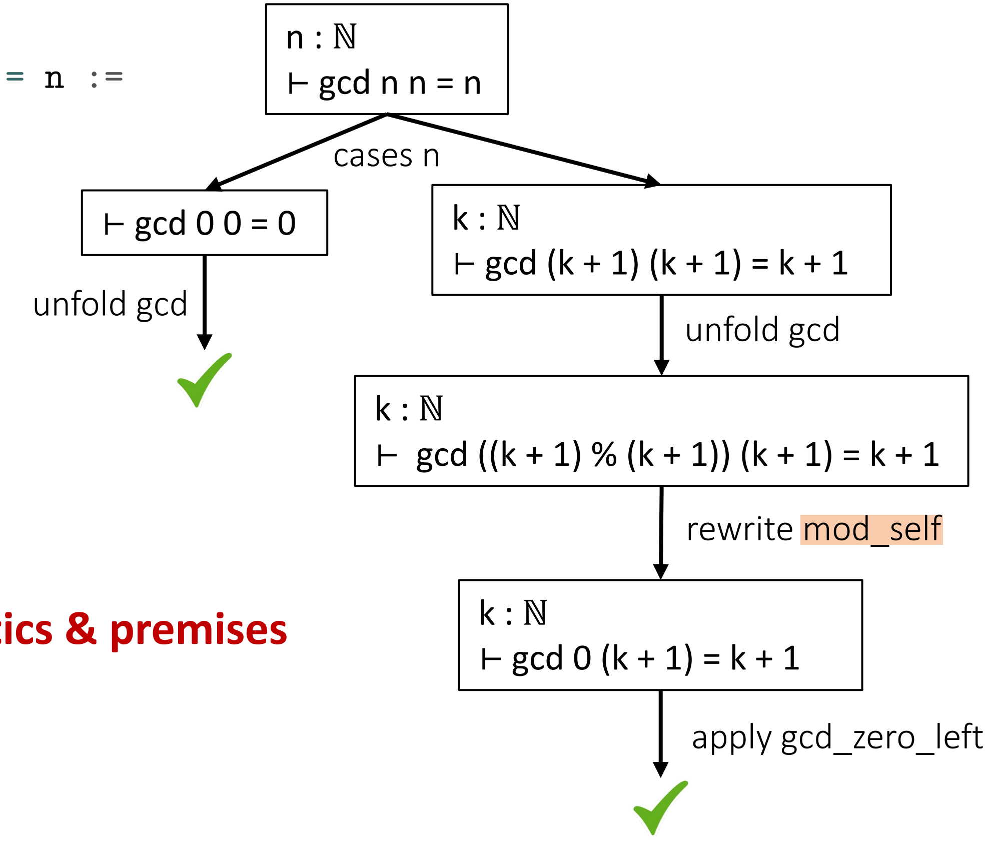


Proof assistant

Theorem Proving in Proof Assistants

Human

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

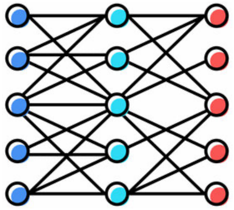


Proof assistant

- **Bottleneck: Finding the right tactics & premises**

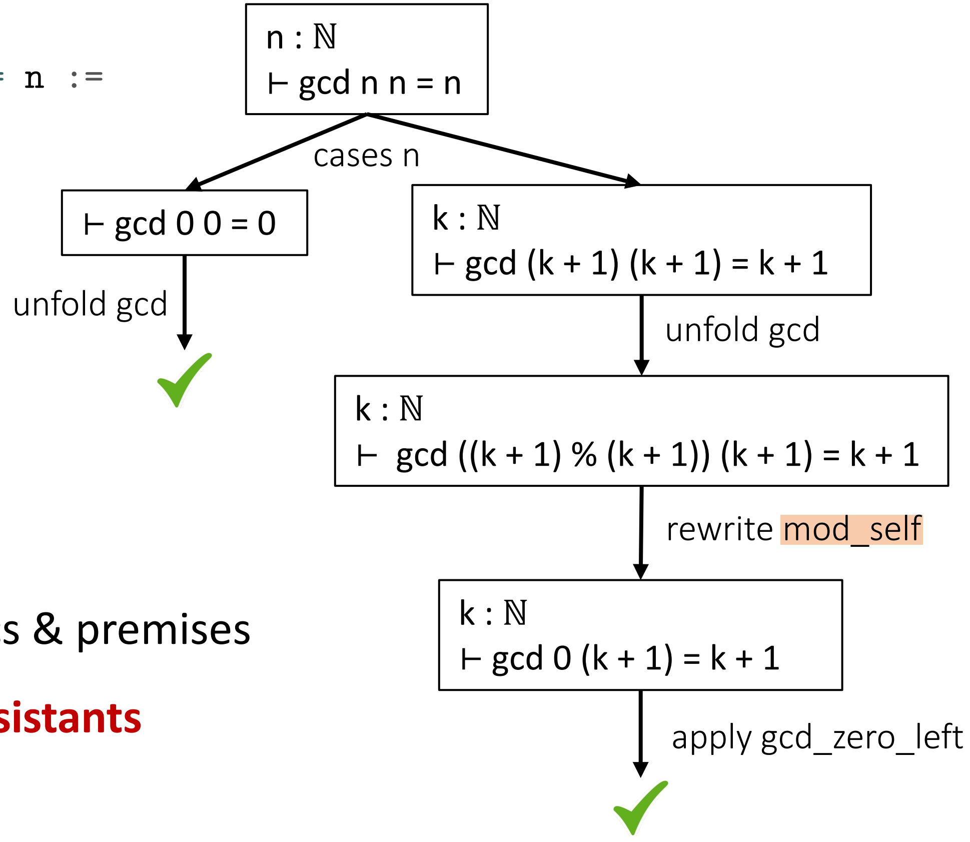
Theorem Proving in Proof Assistants

Human



Machine learning

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

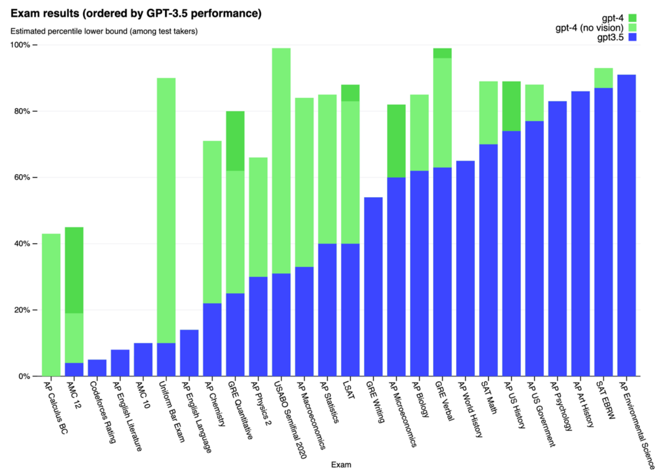


Proof assistant

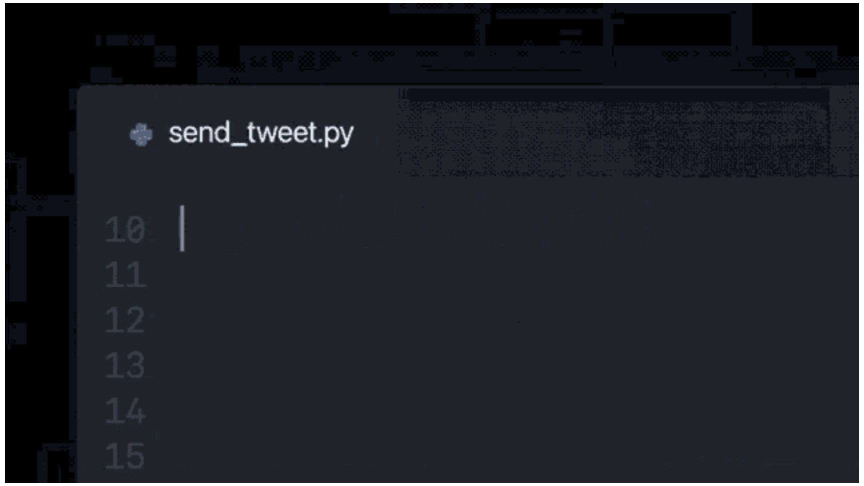
- Bottleneck: Finding the right tactics & premises
- **Learning to interact with proof assistants**

Large Language Models (LLMs)

- Very big neural networks, massive data, predicting the next word
- Good at elementary math and coding



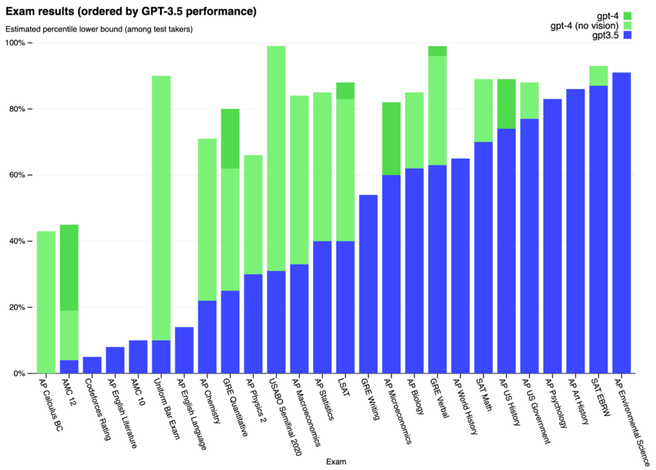
GPT-4 on standard exams (SAT, LSAT, etc.)



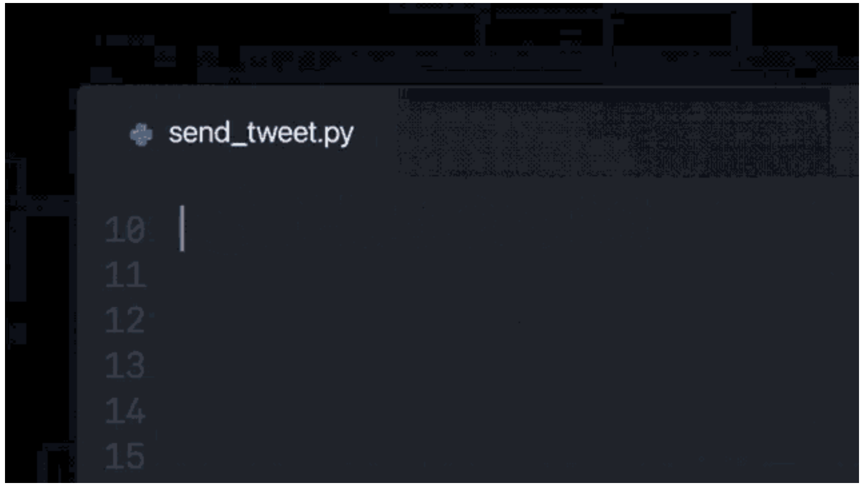
GitHub Copilot

Large Language Models (LLMs)

- Very big neural networks, massive data, predicting the next word
- Good at elementary math and coding



GPT-4 on standard exams (SAT, LSAT, etc.)



GitHub Copilot

- **LLMs are potentially powerful tools for theorem proving**

Theorem Proving as a Challenge for LLMs

- Advanced mathematical reasoning
    - Bigger models are not sufficient
    - May need formal representations in proof assistants

- Rigorous evaluation w/o hallucination
    - LLMs are hard to evaluate
    - LLMs tend to hallucinate
    - Relatively easy to check if formal proofs are correct

LLMs for Theorem Proving: Existing Work and Barriers

OpenAI

Solving (some) formal math olympiad problems

LEAN

Polu and Sutskever, GPT-f, 2020
Han et al., PACT, 2022
Polu et al., 2023
Lample et al., HTPS 2022

Meta AI

Teaching AI advanced mathematical reasoning

November 3, 2022

LLMs for Theorem Proving: Existing Work and Barriers

Jiang et al., LISA, 2021
Jiang et al., Thor, 2022
First et al., Baldur, 2023
Polu and Sutskever, GPT-f, 2020
Han et al., PACT, 2022
Polu et al., 2023
Lample et al., HTPS 2022

LLMs for Theorem Proving: Existing Work and Barriers

Jiang et al., LISA, 2021
Jiang et al., Thor, 2022
First et al., Baldur, 2023
Polu and Sutskever, GPT-f, 2020
Han et al., PACT, 2022
Polu et al., 2023
Lample et al., HTPS 2022
Wang et al., DT-Solver, 2023

LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available
Jiang et al., LISA, 2021	✓
Jiang et al., Thor, 2022	✓
First et al., Baldur, 2023	✗
Polu and Sutskever, GPT-f, 2020	✗
Han et al., PACT, 2022	✗
Polu et al., 2023	✗
Lample et al., HTPS 2022	✗
Wang et al., DT-Solver, 2023	✓

LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available	Model available	Code available
Jiang et al., LISA, 2021	✓	✗	✗
Jiang et al., Thor, 2022	✓	✗	✗
First et al., Baldur, 2023	✓✗	✗	✗
Polu and Sutskever, GPT-f, 2020	✗	✗	✗
Han et al., PACT, 2022	✓✗	✗	✗
Polu et al., 2023	✓✗	✗	✗
Lample et al., HTPS 2022	✓✗	✗	✗
Wang et al., DT-Solver, 2023	✓	✗	✗

LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available	Model available	Code available	Interaction tool available
Jiang et al., LISA, 2021	✓	✗	✗	✓
Jiang et al., Thor, 2022	✓	✗	✗	✓
First et al., Baldur, 2023	✓✗	✗	✗	✓
Polu and Sutskever, GPT-f, 2020	✗	✗	✗	✗
Han et al., PACT, 2022	✓✗	✗	✗	✓
Polu et al., 2023	✓✗	✗	✗	✓
Lample et al., HTPS 2022	✓✗	✗	✗	✗
Wang et al., DT-Solver, 2023	✓	✗	✗	✗

LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available	Model available	Code available	Interaction tool available	Model size (# params)	Compute (hours)
Jiang et al., LISA, 2021	✓	✗	✗	✓	163M	-
Jiang et al., Thor, 2022	✓	✗	✗	✓	700M	1K on TPU
First et al., Baldur, 2023	✓✗	✗	✗	✓	62,000M	-
Polu and Sutskever, GPT-f, 2020	✗	✗	✗	✗	774M	40K on GPU
Han et al., PACT, 2022	✓✗	✗	✗	✓	837M	1.5K on GPU
Polu et al., 2023	✓✗	✗	✗	✓	774M	48K on GPU
Lample et al., HTPS 2022	✓✗	✗	✗	✗	600M	34K on GPU
Wang et al., DT-Solver, 2023	✓	✗	✗	✗	774M	1K on GPU

LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available	Model available	Code available	Interaction tool available	Model size (# params)	Compute (hours)
Jiang et al., LISA, 2021	✓	✗	✗	✓	163M	-
Jiang et al., Thor, 2022	✓	✗	✗	✓	700M	1K on TPU
First et al., Baldur, 2023	✓✗	✗	✗	✓	62,000M	-
Polu and Sutskever, GPT-f, 2020	✗	✗	✗	✗	774M	40K on GPU
Han et al., PACT, 2022	✓✗	✗	✗	✓	837M	1.5K on GPU
Polu et al., 2023	✓✗	✗	✗	✓	774M	48K on GPU
Lample et al., HTPS 2022	✓✗	✗	✗	✗	600M	34K on GPU
Wang et al., DT-Solver, 2023	✓	✗	✗	✗	774M	1K on GPU
LeanDojo (ours)	✓	✓	✓	✓	**517M**	**120 on GPU**

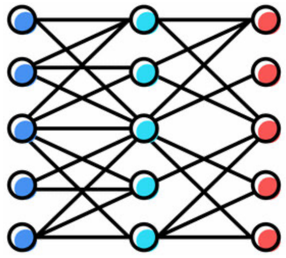
LLMs for Theorem Proving: Existing Work and Barriers

	Dataset available	Model available	Code available	Interaction tool available	Model size (# params)	Compute (hours)
Jiang et al., LISA, 2021	✓	✗	✗	✓	163M	-
Jiang et al., Thor, 2022	✓	✗	✗	✓	700M	1K on TPU
First et al., Baldur, 2023	✓✗	✗	✗	✓	62,000M	-
Polu and Sutskever, GPT-f, 2020	✗	✗	✗	✗	774M	40K on GPU
Han et al., PACT, 2022	✓✗	✗	✗	✓	837M	1.5K on GPU
Polu et al., 2023	✓✗	✗	✗	✓	774M	48K on GPU
Lample et al., HTPS 2022	✓✗	✗	✗	✗	600M	34K on GPU
Wang et al., DT-Solver, 2023	✓	✗	✗	✗	774M	1K on GPU
LeanDojo (ours)	✓	✓	✓	✓	**517M**	**120 on GPU**

Give researchers access to state-of-the-art LLM-based provers with modest computational costs

LeanDojo

Lean (Lean 3 or Lean 4)



Machine learning model

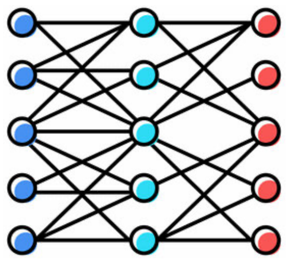
LeanDojo

Data extraction

LeanDojo Benchmark

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

Lean



Machine learning model

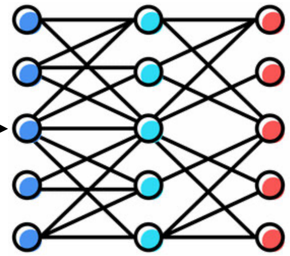
LeanDojo

Data extraction →

LeanDojo Benchmark

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

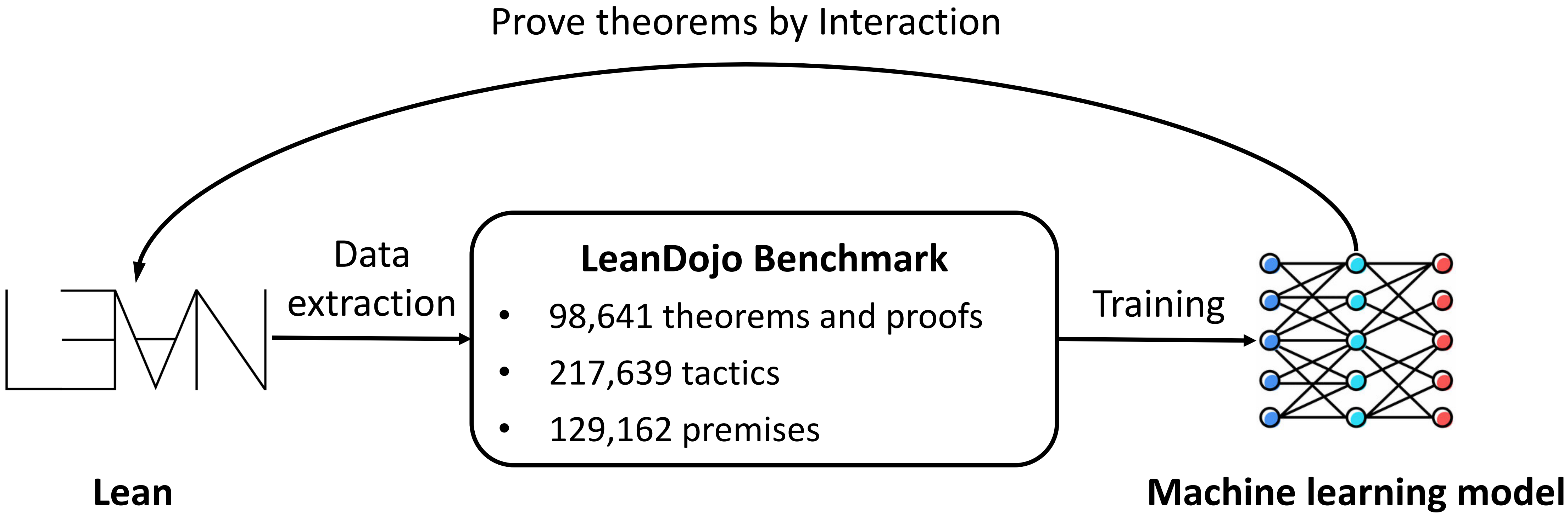
Training →



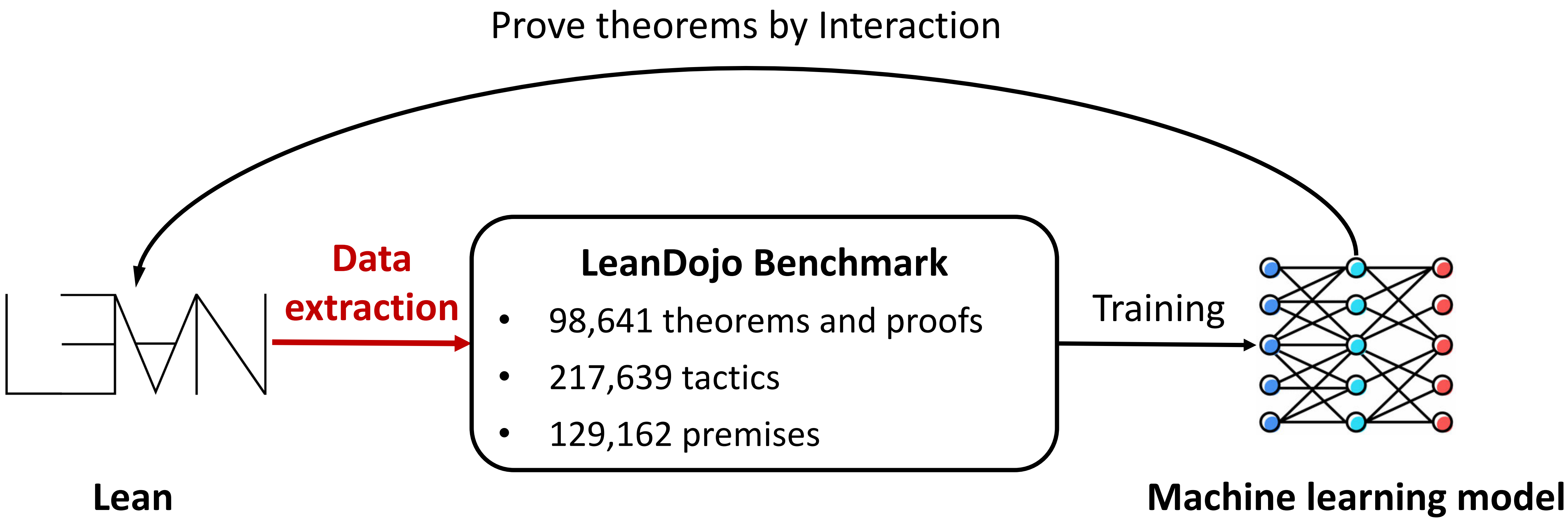
Lean

Machine learning model

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

LeanDojo

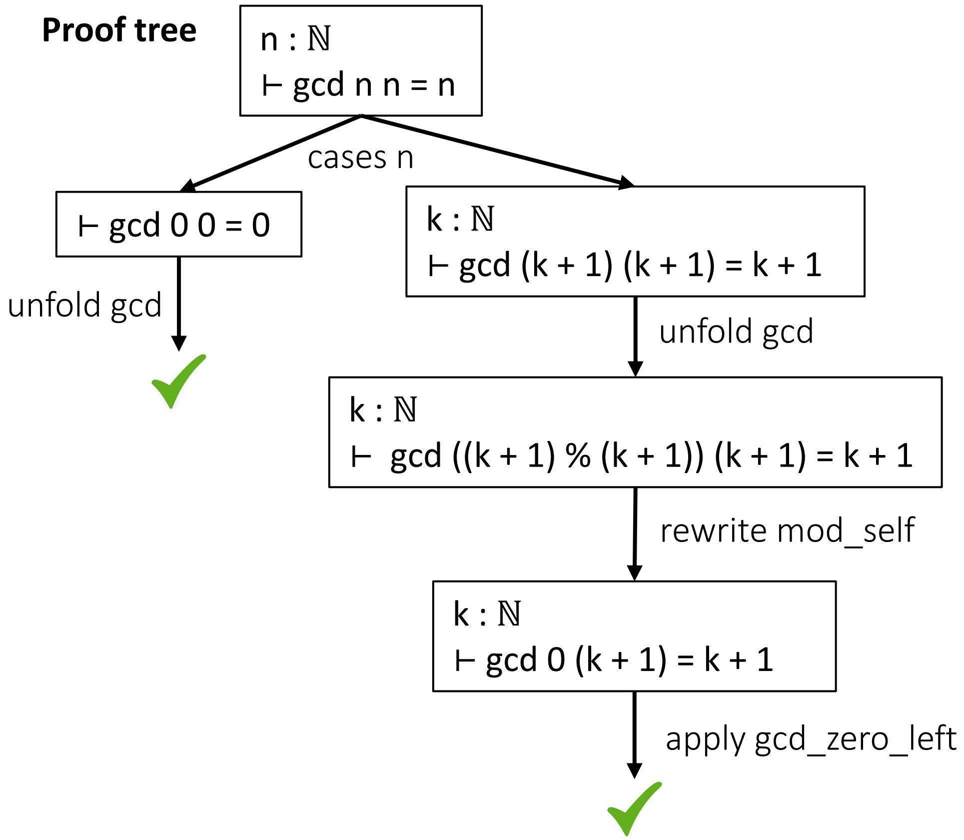


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

Extracting States and Tactics

- Need **(state, tactic)** pairs for training
  - Tactics could be obtained by parsing the Lean source code into ASTs
  - Proof states are not available in the code

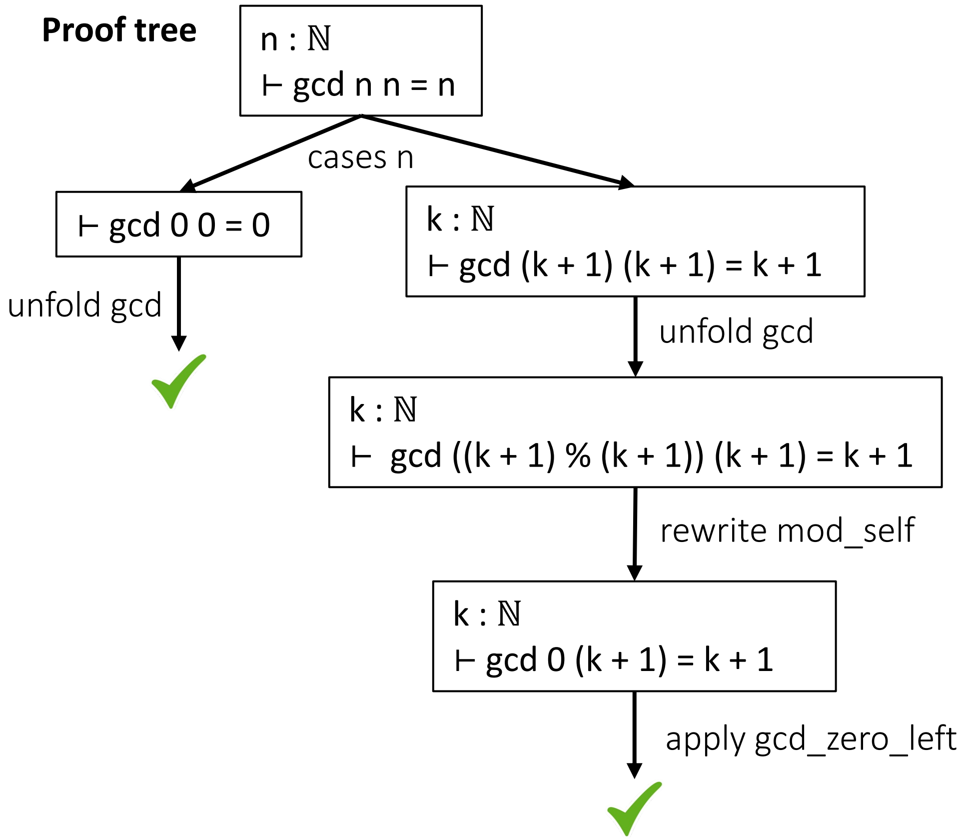
```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```



Extracting Premises in the Same File

- Tactics rely on **premises**
  - Lemmas
  - Definitions

```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

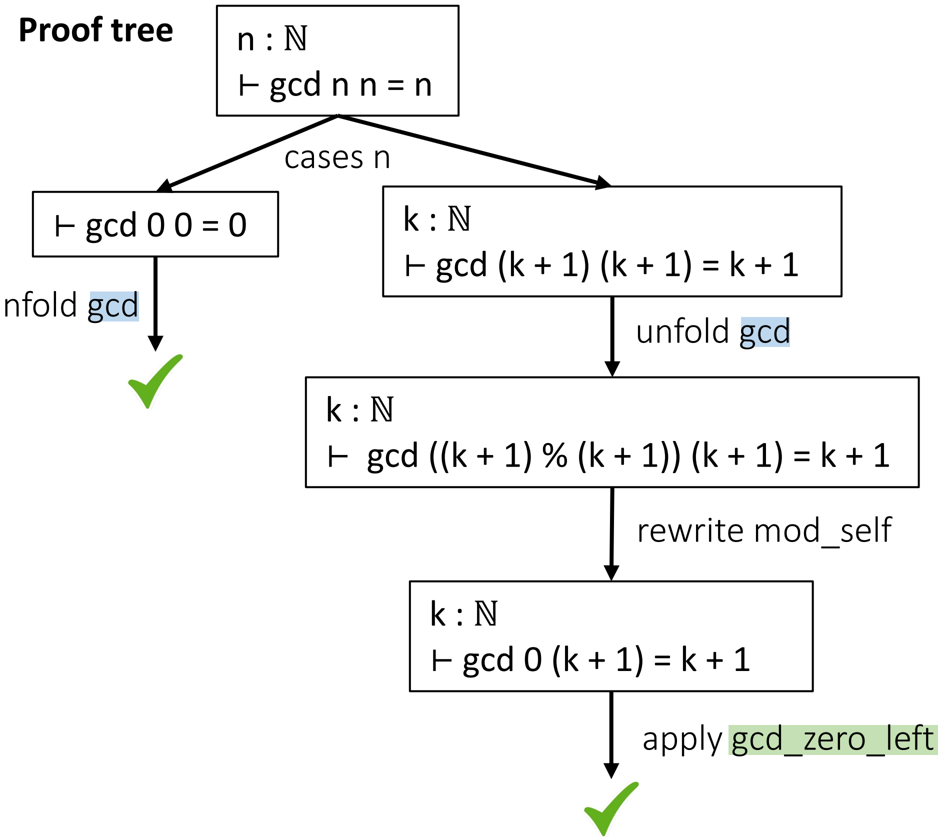


Extracting Premises in the Same File

- Tactics rely on **premises**
  - Lemmas
  - Definitions

data/nat/gcd.lean

```
def gcd : nat → nat → nat                          -- gcd z y
| 0        y := y                                  -- Case 1: z == 0
| (x + 1) y := gcd (y % (x + 1)) (x + 1)           -- Case 2: z > 0

theorem gcd_zero_left (x : nat) : gcd 0 x = x := begin simp [gcd] end

theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```



Extracting Premises from Other Files

Math library

data/nat/lemmas.lean

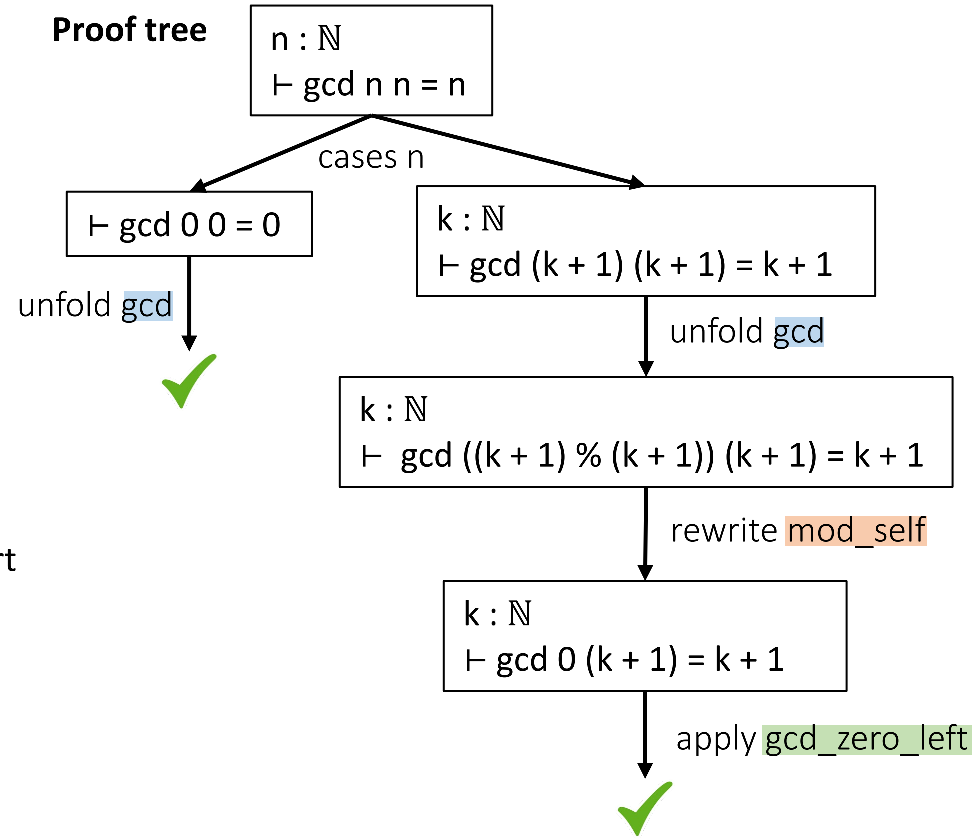
```
theorem mod_self (n : nat) : n % n = 0 :=
begin
  rw [mod_eq_sub_mod (le_refl _), nat.sub_self, zero_mod]
end
```

Import

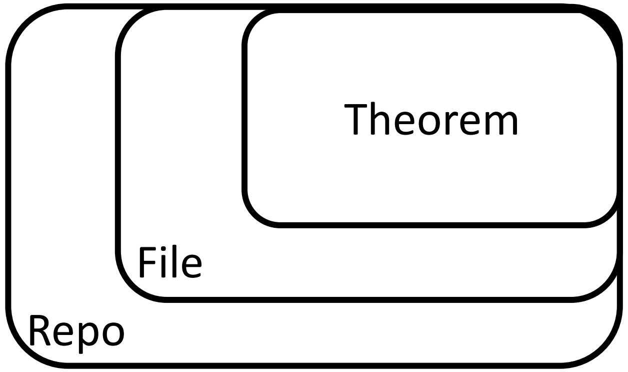
data/nat/gcd.lean

```
def gcd : nat → nat → nat                    -- gcd z y
| 0        y := y                            -- Case 1: z == 0
| (x + 1) y := gcd (y % (x + 1)) (x + 1)    -- Case 2: z > 0

theorem gcd_zero_left (x : nat) : gcd 0 x = x := begin simp [gcd] end

theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```

Proof tree

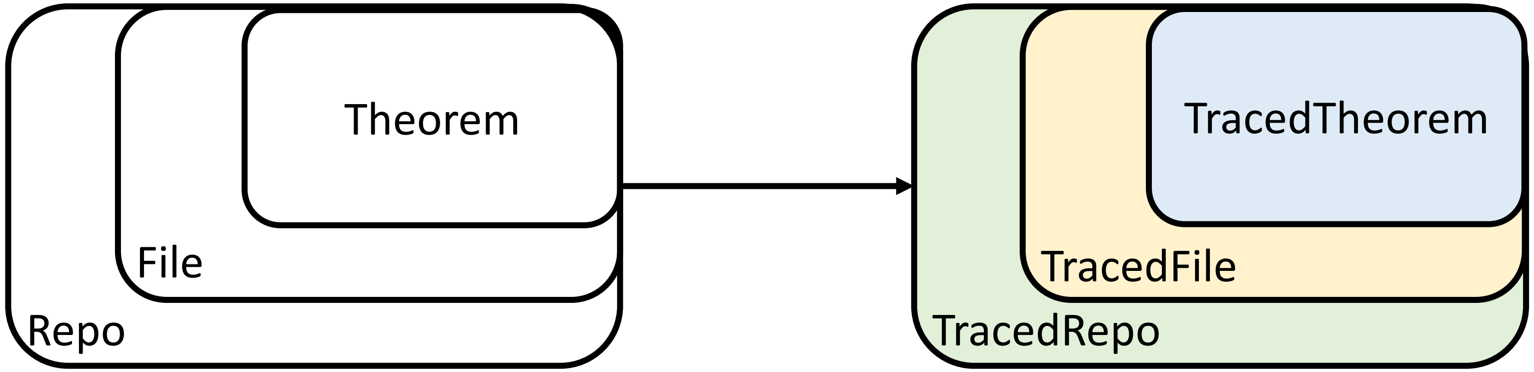


Data Extraction in LeanDojo



Arbitrary Lean repo on GitHub

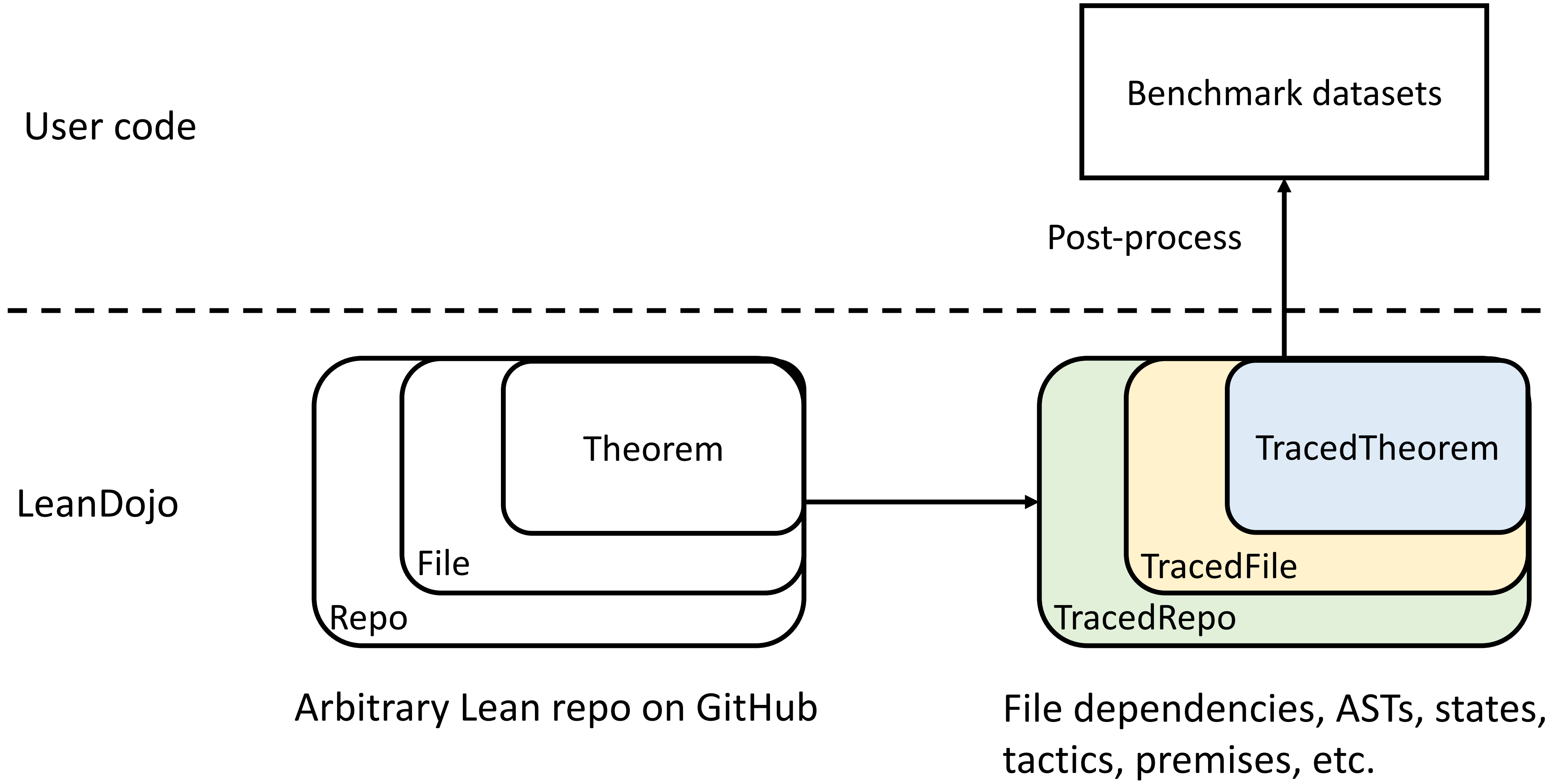
Data Extraction in LeanDojo



Arbitrary Lean repo on GitHub

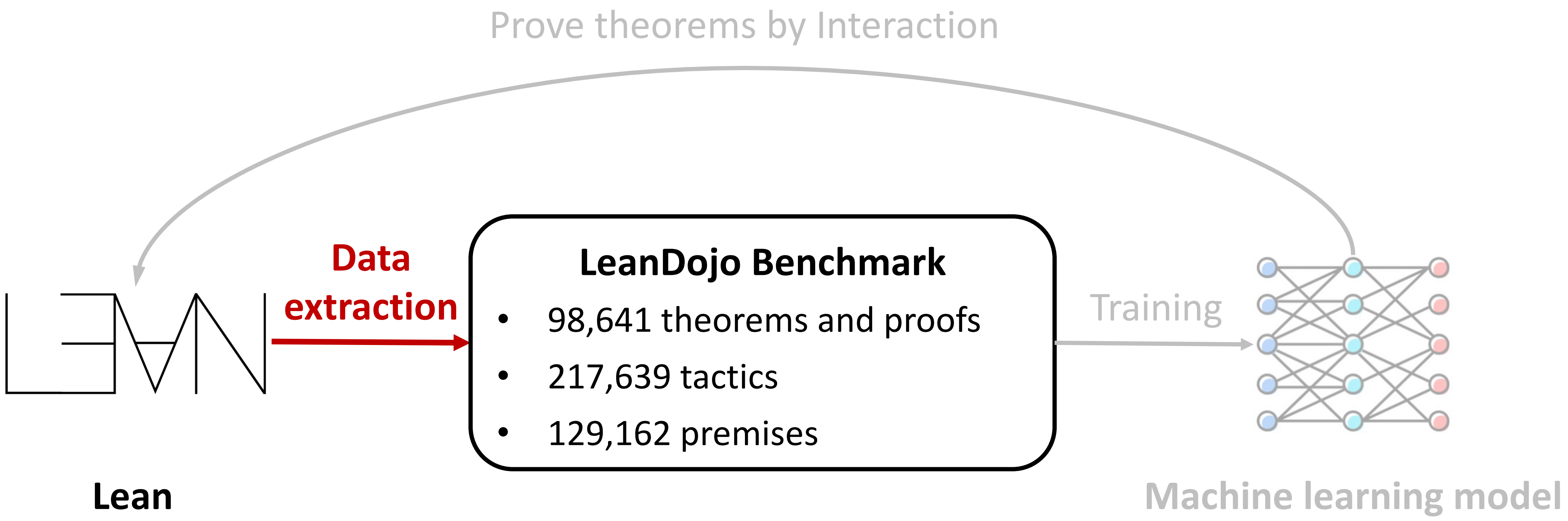
File dependencies, ASTs, states, tactics, premises, etc.

Data Extraction in LeanDojo

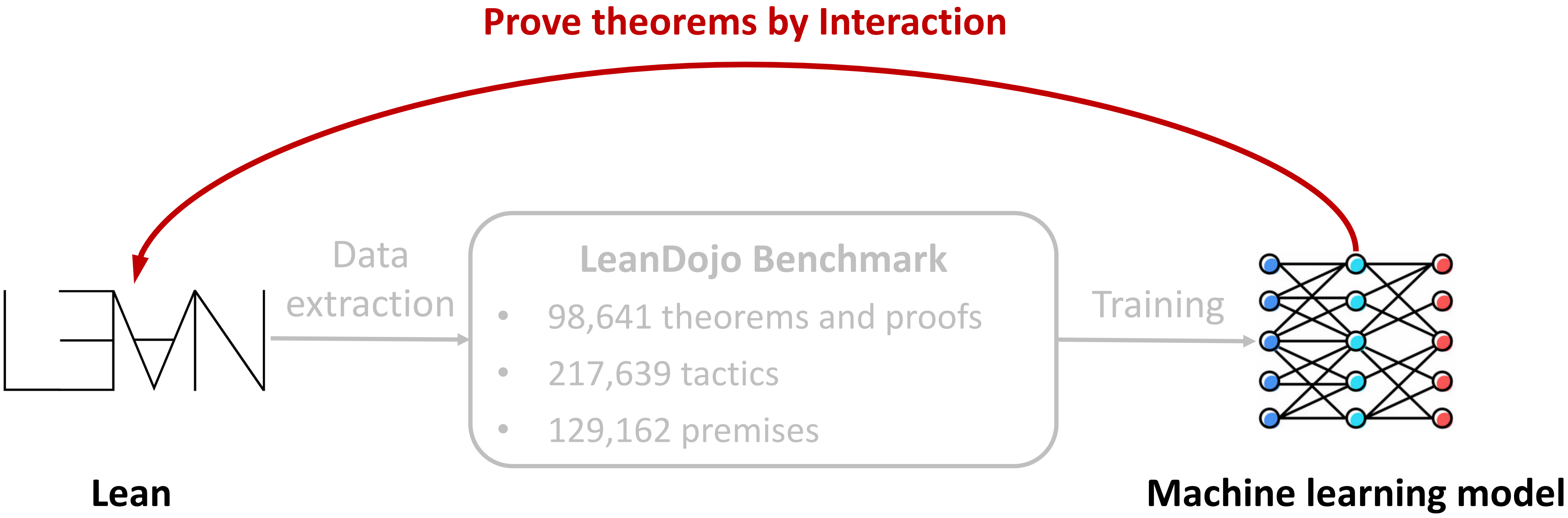


Data Extraction Demo

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

LeanDojo


Prove theorems by Interaction
Data
extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

Interacting with Lean Programmatically

- An interface for the model to observe states and run tactics
  - initialize(theorem): Given a theorem, return its initial state
  - run_tac(state, tactic): Run a tactic on a given state and return the next state

- [Demo]()

Interacting with Lean Programmatically

- An interface for the model to observe states and run tactics
  - initialize(theorem): Given a theorem, return its initial state
  - run_tac(state, tactic): Run a tactic on a given state and return the next state

- [Demo]

- **The first tool to interact with Lean 3 reliably**
  - Existing tool, lean-gym, misjudges 21% correct proofs as incorrect
  - Only 2.1% for LeanDojo

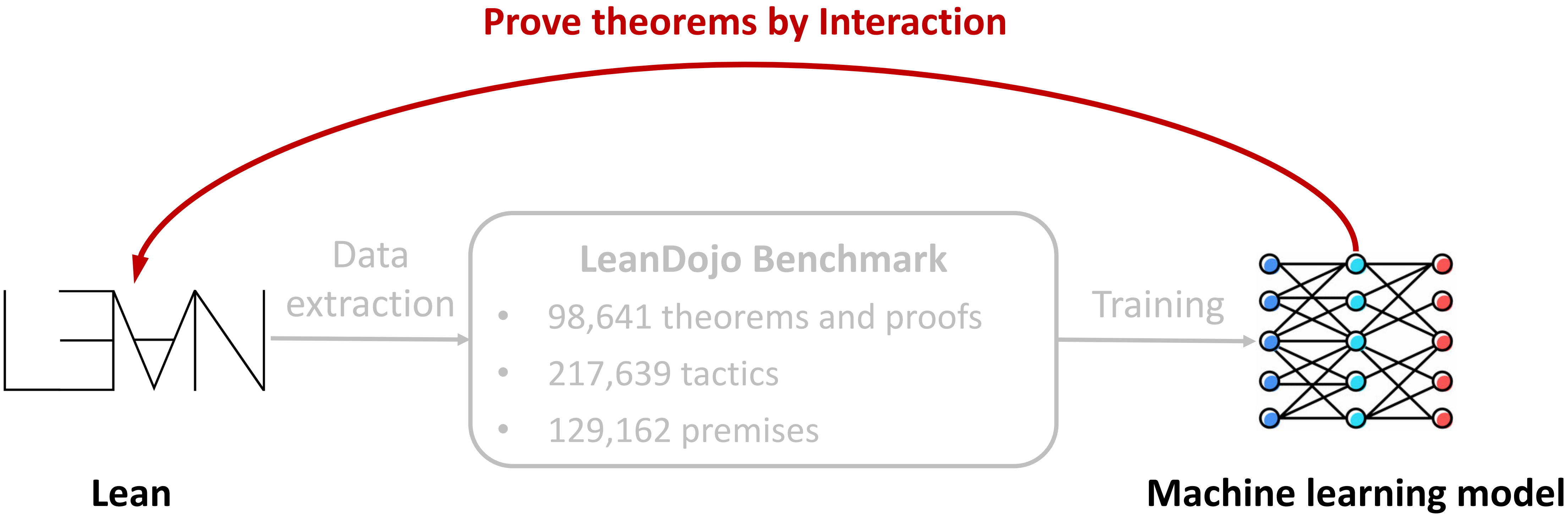
Interacting with Lean Programmatically

- An interface for the model to observe states and run tactics
  - initialize(theorem): Given a theorem, return its initial state
  - run_tac(state, tactic): Run a tactic on a given state and return the next state

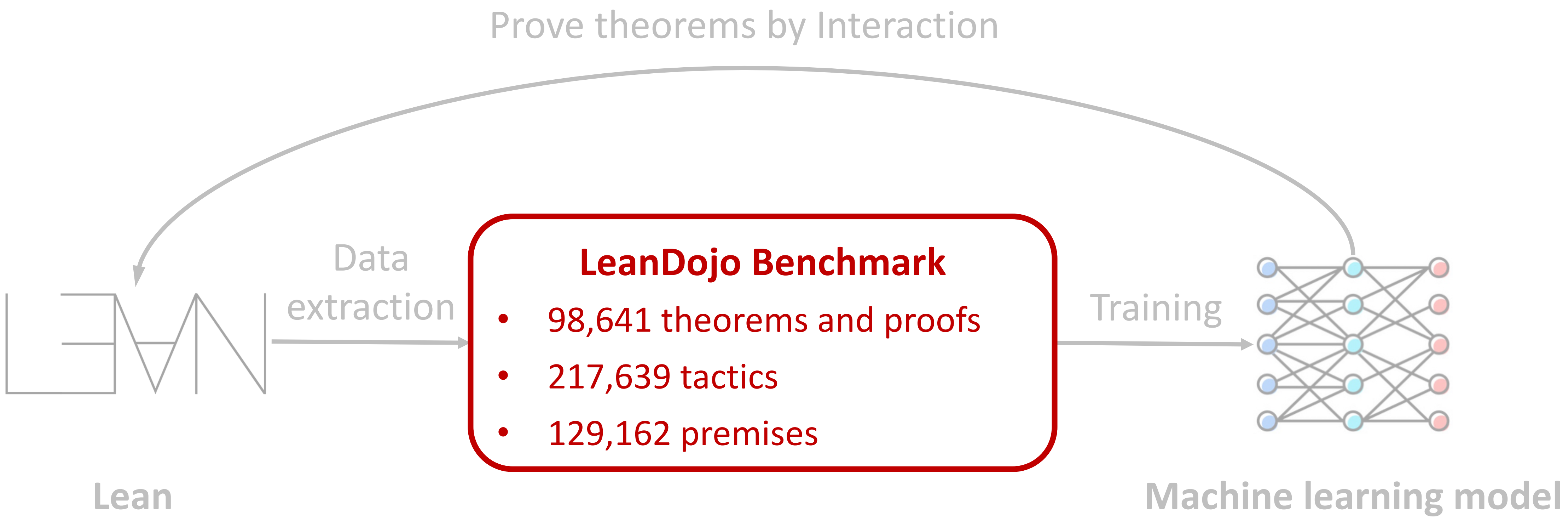
- Demo

- The first tool to interact with Lean 3 reliably
  - Existing tool, lean-gym, misjudges 21% correct proofs as incorrect
  - Only 2.1% for LeanDojo
- **The first tool to interact with Lean 4**
  - Several prototypes and ongoing projects, no mature tool before LeanDojo

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
98,641 theorems and proofs
217,639 tactics
129,162 premises
Training
Lean
Machine learning model

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

Constructing Benchmarks using LeanDojo

- **LeanDojo Benchmark**, from mathlib on Aug 5, 2023
  - 98,641 theorems and proofs
  - 217,639 tactics
  - 129,162 premises
- **LeanDojo Benchmark 4**, from mathlib4 on Aug 10, 2023
  - 100,780 theorems and proofs
  - 209,133 tactics
  - 101,500 premises
- Easy to construct your own benchmarks

Challenging Data Split

- random: Splitting theorems into training/validation/testing randomly
- LLMs can prove seemingly nontrivial theorems by memorizing similar proofs in training

Challenging Data Split

- random: Splitting theorems into training/validation/testing randomly
- LLMs can prove seemingly nontrivial theorems by memorizing similar proofs in training

src/algebra/quaternion.lean

```
lemma conj_mul : (a * b).conj = b.conj * a.conj := begin
  ext; simp; ring_exp
end

lemma conj_conj_mul : (a.conj * b).conj = b.conj * a := begin
  rw [conj_mul, conj_conj]
end

lemma conj_mul_conj : (a * b.conj).conj = b * a.conj := begin
  rw [conj_mul, conj_conj]
end
```

Challenging Data Split

- random: Splitting theorems into training/validation/testing randomly
- LLMs can prove seemingly nontrivial theorems by memorizing similar proofs in training

src/algebra/quaternion.lean

```
lemma conj_mul : (a * b).conj = b.conj * a.conj := begin
  ext; simp; ring_exp
end
```

Train

```
lemma conj_conj_mul : (a.conj * b).conj = b.conj * a := begin
  rw [conj_mul, conj_conj]
end
```

Test

```
lemma conj_mul_conj : (a * b.conj).conj = b * a.conj := begin
  rw [conj_mul, conj_conj]
end
```

Challenging Data Split

- random: Splitting theorems into training/validation/testing randomly
- LLMs can prove seemingly nontrivial theorems by memorizing similar proofs in training

src/algebra/quaternion.lean

```
lemma conj_mul : (a * b).conj = b.conj * a.conj := begin
  ext; simp; ring_exp
end
```

Train

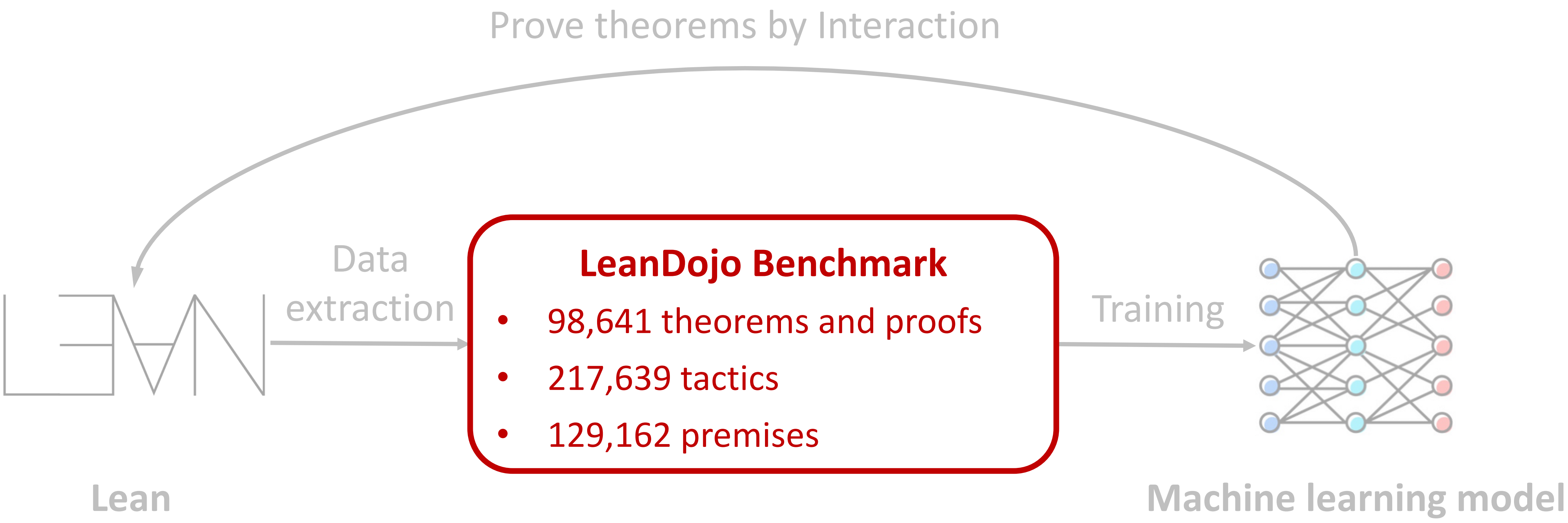
```
lemma conj_conj_mul : (a.conj * b).conj = b.conj * a := begin
  rw [conj_mul, conj_conj]
end
```

Test

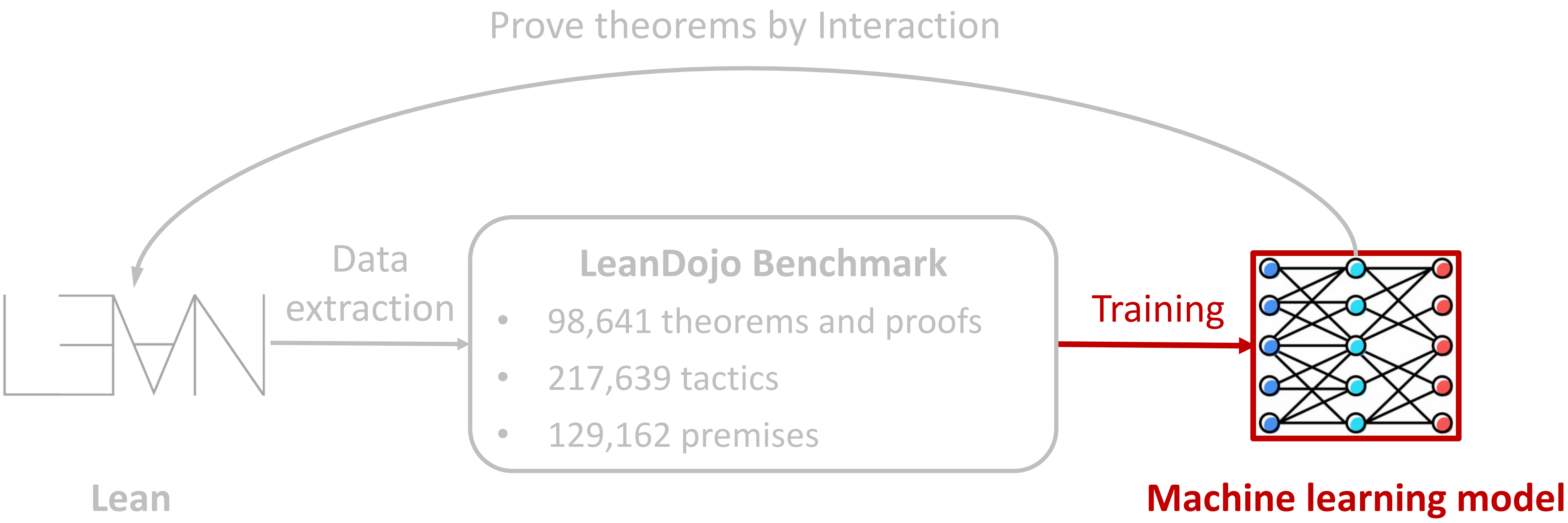
```
lemma conj_mul_conj : (a * b.conj).conj = b * a.conj := begin
  rw [conj_mul, conj_conj]
end
```

- novel_premises: **Testing proofs must use >1 premise that is never used in training**

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

LeanDojo

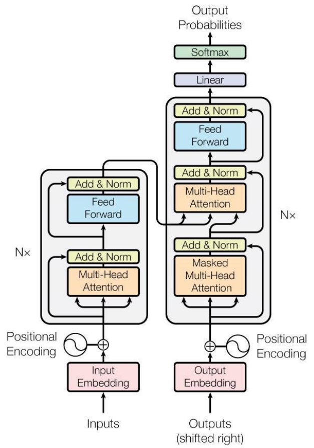

Prove theorems by Interaction
Data
extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

Tactic Generator in Existing LLM-Based Provers

- State -> tactic
- The model can use premises only by memorizing their names

State

k : ℕ
⊢ gcd ((k + 1) % (k + 1)) (k + 1) = k + 1



→ rewrite mod_self

Tactic

[Vaswani et al., NeurIPS 2017]

Retrieval-Augmented Prover (ReProver)

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

State

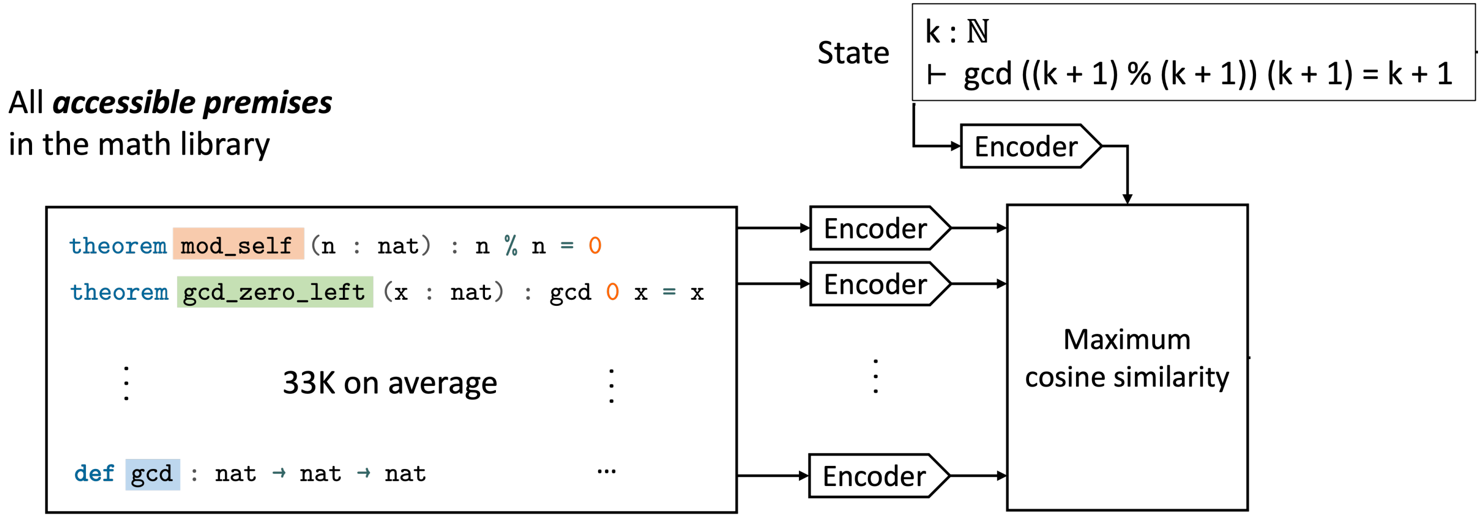
k : ℕ
⊢ gcd ((k + 1) % (k + 1)) (k + 1) = k + 1

All ***accessible premises***
in the math library

```
theorem mod_self (n : nat) : n % n = 0
theorem gcd_zero_left (x : nat) : gcd 0 x = x
   ⋮            33K on average            ⋮
def gcd : nat → nat → nat                 …
```

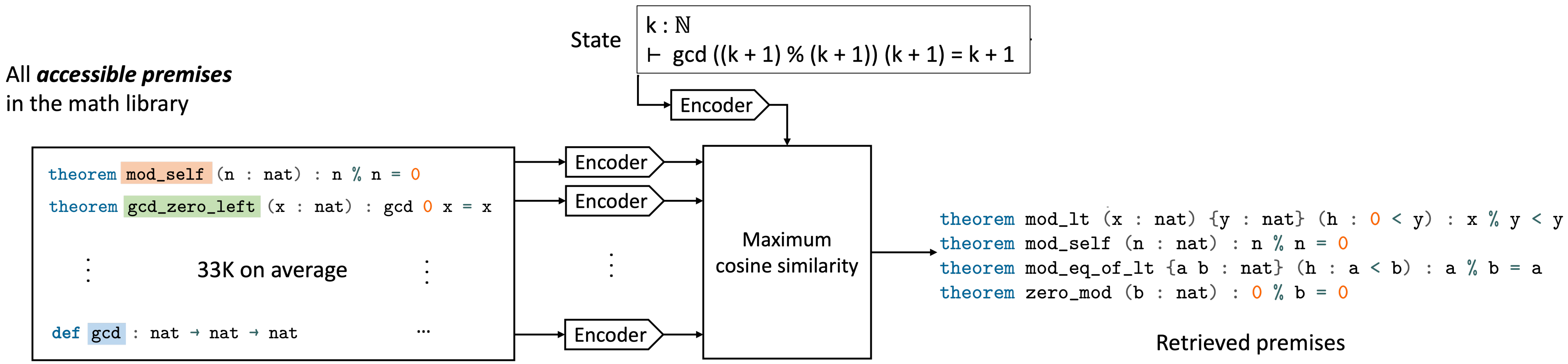
Retrieval-Augmented Prover (ReProver)

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



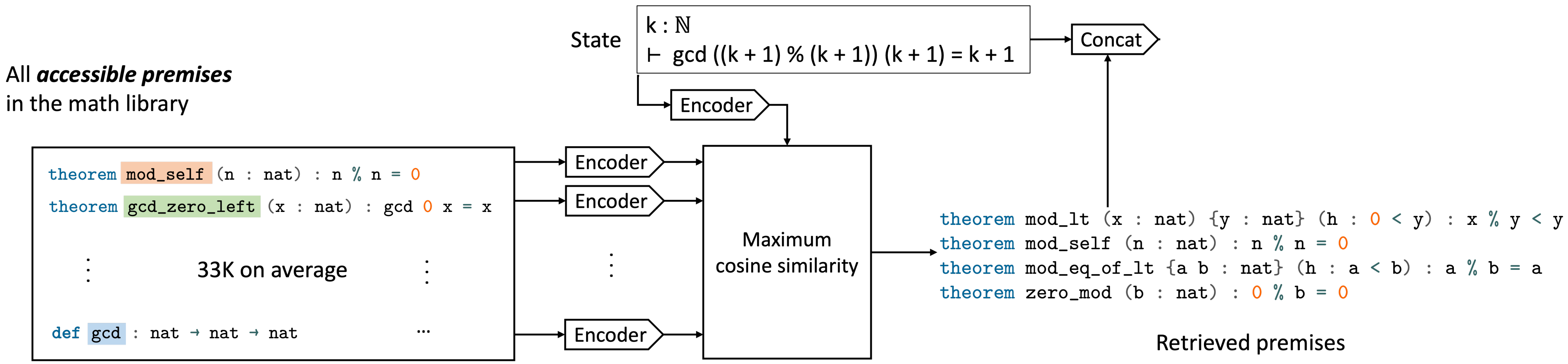
Retrieval-Augmented Prover (ReProver)

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



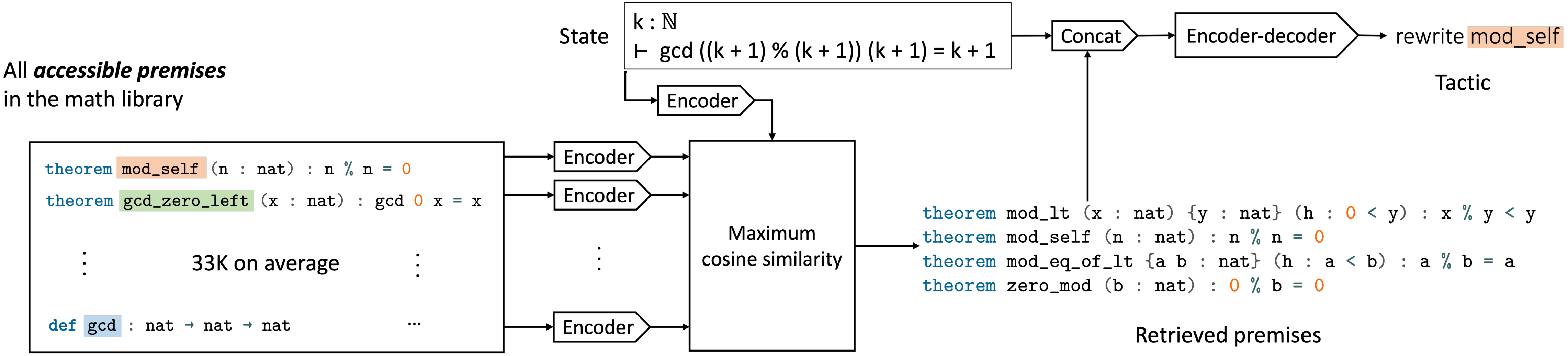
Retrieval-Augmented Prover (ReProver)

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

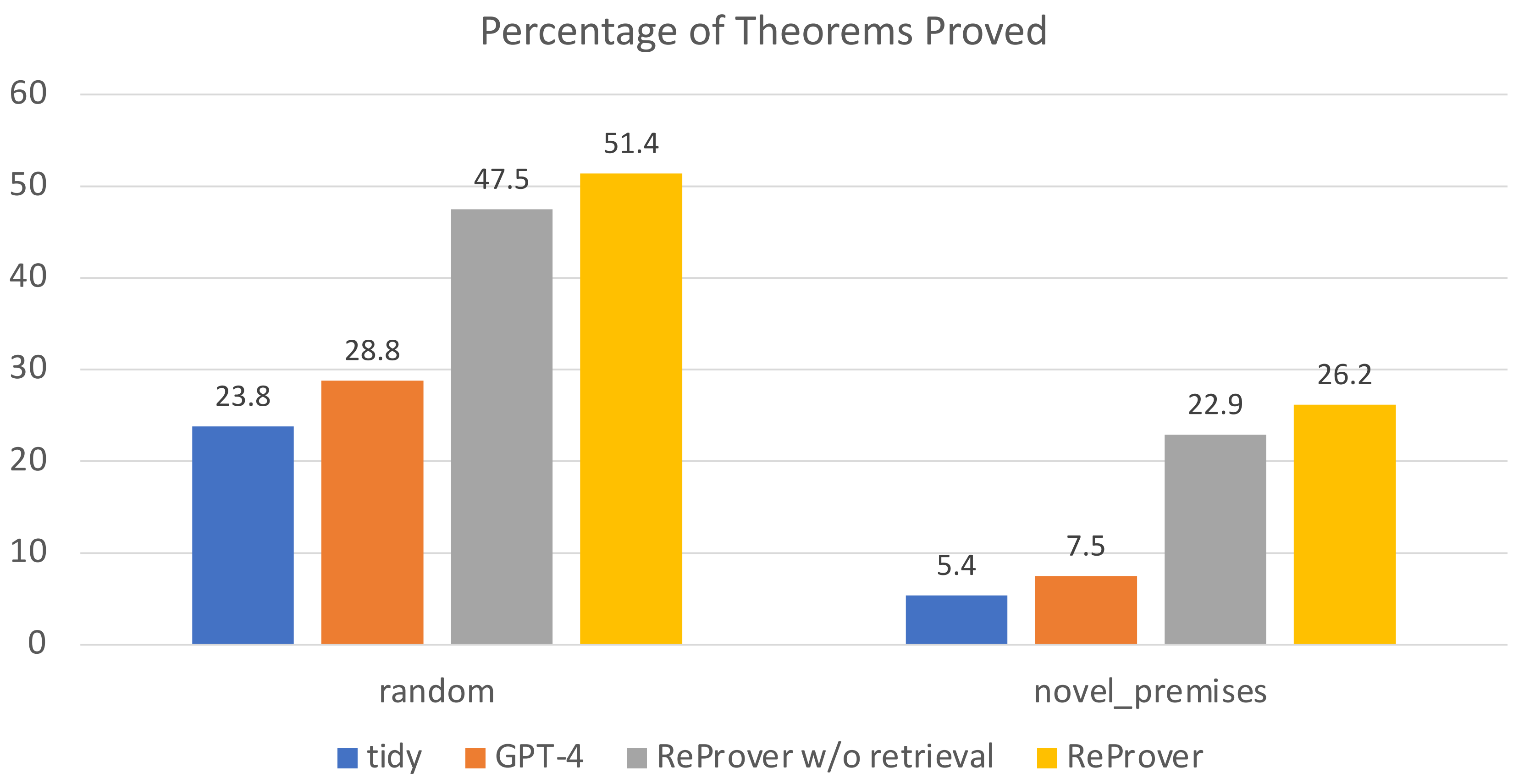


Retrieval-Augmented Prover (ReProver)

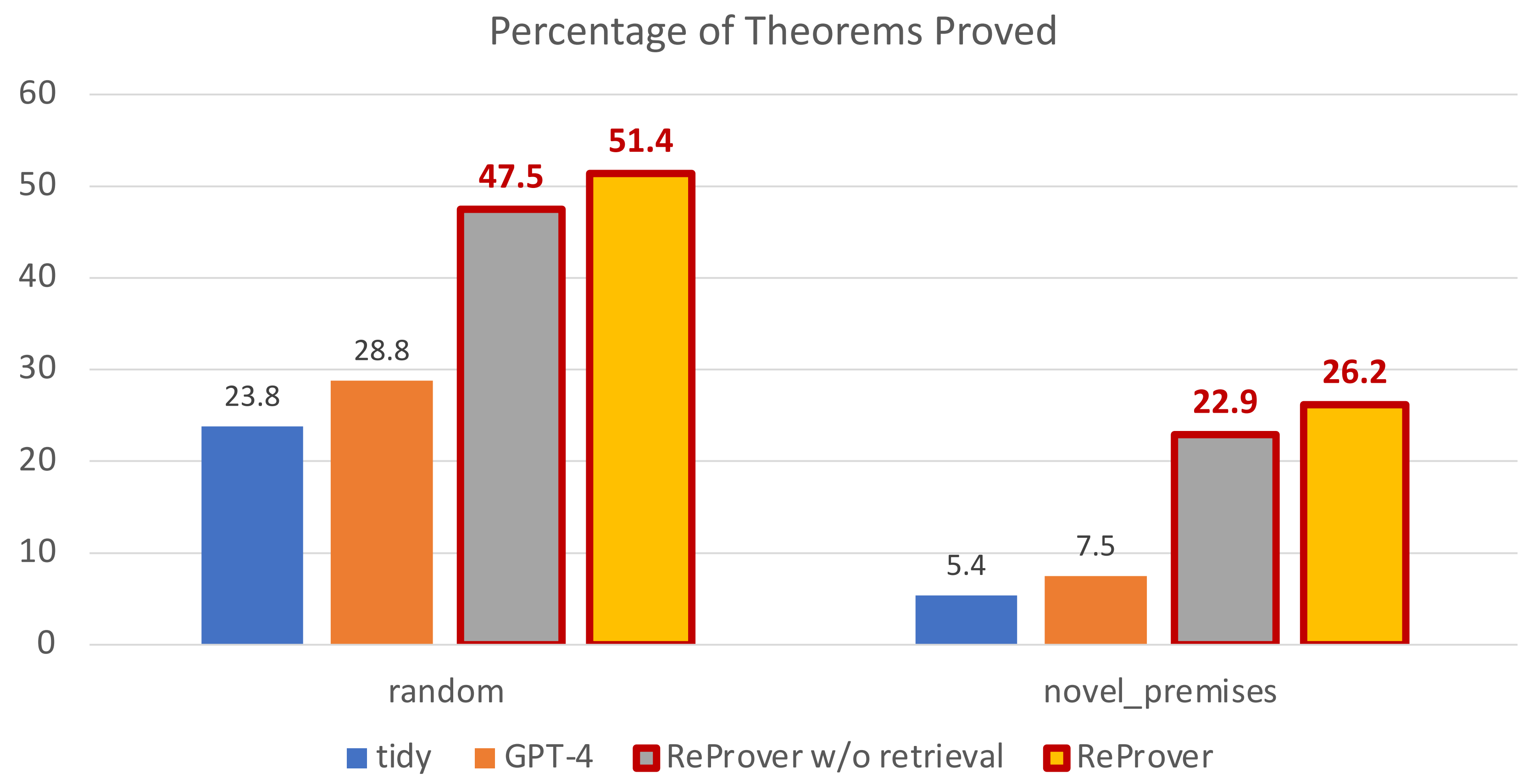
- Given a state, we retrieve premises from the set of **all accessible premises**
- Retrieved premises are concatenated with the state and used for tactic generation



Premise Retrieval Improves Theorem Proving



Premise Retrieval Improves Theorem Proving



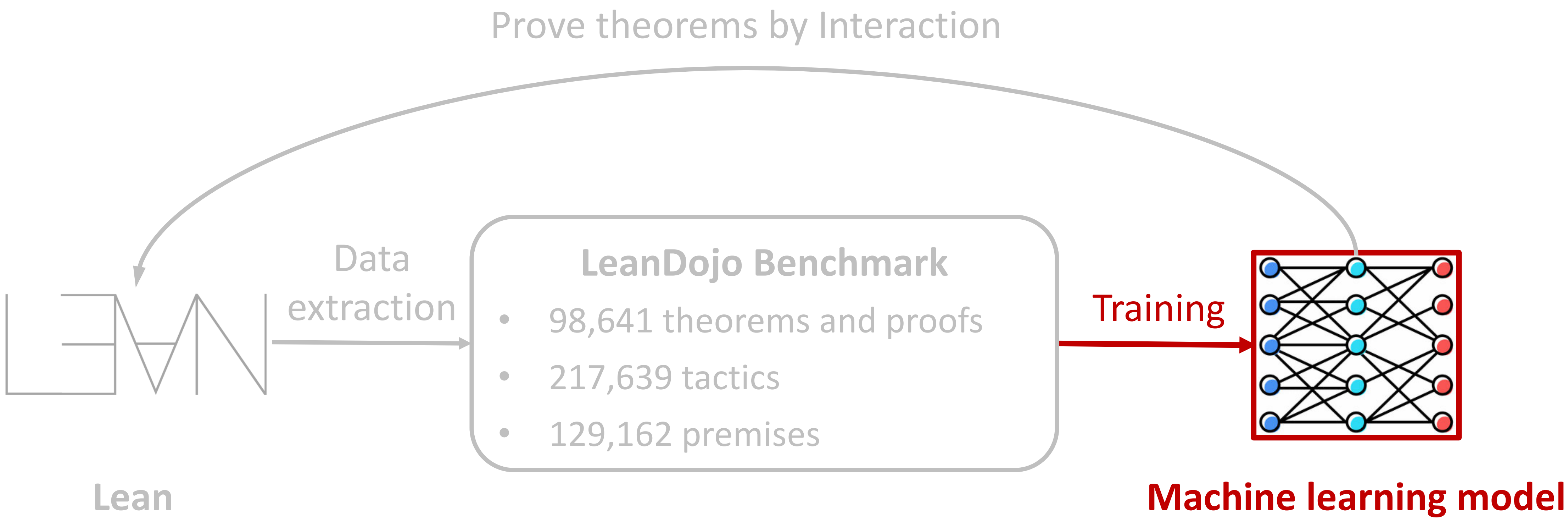
Discovering New Proofs on MiniF2F and ProofNet

- We evaluate the model on MiniF2F and ProofNet (in zero shot) to discover new Lean proofs

```
theorem exercise_2_3_2 {G : Type*} [group G] (a b : G) :
   g : G, b * a = g * a * b * g ¹ :=
begin
  exact  b, by simp ,
end

theorem exercise_11_2_13 (a b :  ) :
  (of_int a : gaussian_int)   of_int b → a   b :=
begin
  contrapose,
  simp,
end

theorem exercise_1_1_17 {G : Type*} [group G] {x : G} {n :  }
  (hxn: order_of x = n) :
  x ¹ = x ^ (n - 1 :  ) :=
begin
  rw zpow_sub_one,
  simp,
  rw [← hxn, pow_order_of_eq_one],
end
```

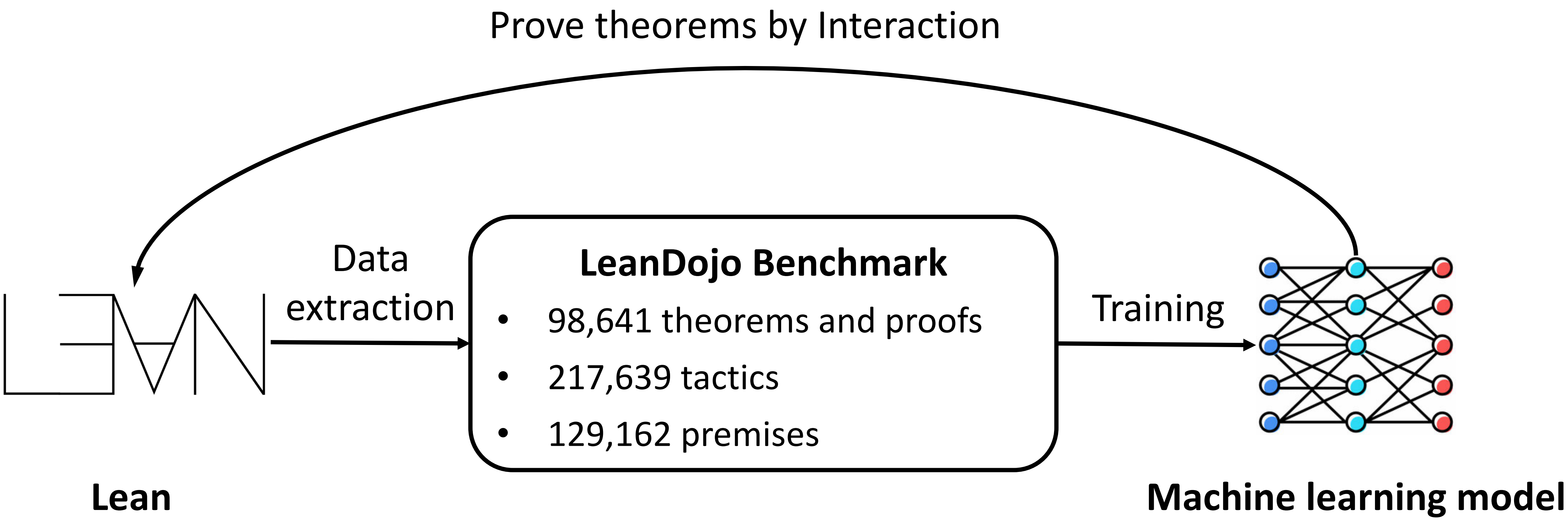
```
theorem exercise_3_1_22b {G : Type*} [group G] (I : Type*)
  (H : I → subgroup G) (hH :   i : I, subgroup.normal (H i)) :
  subgroup.normal (  (i : I), H i):=
begin
  rw infi,
  rw ←set.image_univ,
  rw Inf_image,
  simp [hH],
  haveI :=   i, (H i).normal,
  split,
  intros x hx g,
  rw subgroup.mem_infi at hx  ,
  intro i,
  apply (hH i).conj_mem _ (hx i),
end

theorem exercise_3_4_5a {G : Type*} [group G]
  (H : subgroup G) [is_solvable G] : is_solvable H :=
begin
  apply_instance,
end
```

LeanDojo


Prove theorems by Interaction
Data
extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model

LeanDojo

- https://leandojo.org
- Tutorial @ NeurIPS 2023

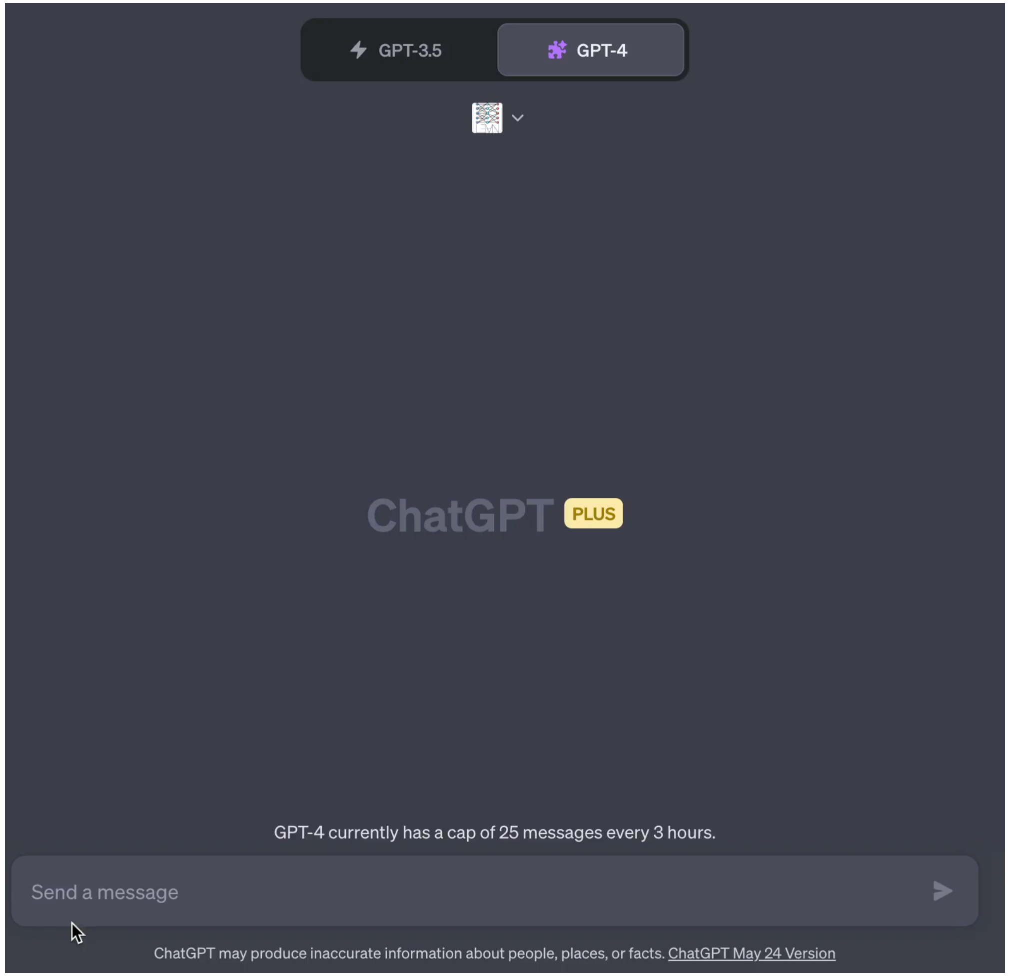


Future Direction: GPT for Theorem Proving in Lean

- ChatGPT can use LeanDojo to interact with Lean

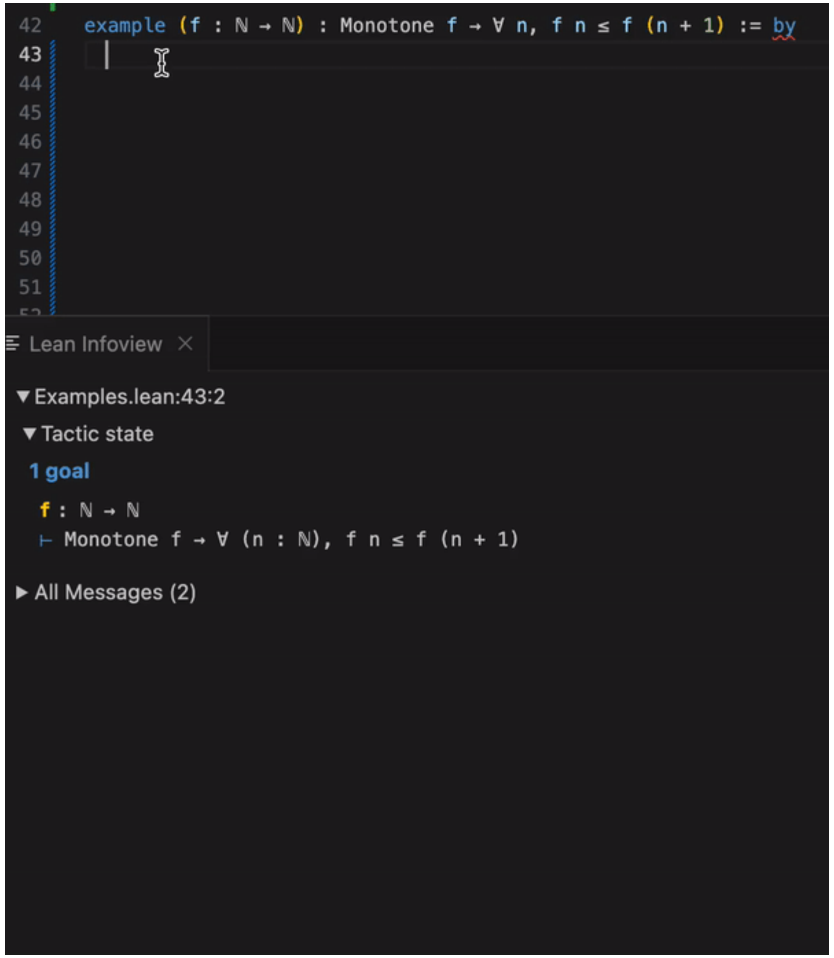
- Strengths:
  - Interleave informal math with formal proofs
  - Explain and correct errors
  - Steerable via prompt engineering

- Limitations:
  - Hallucinating informal math
  - Unable to prove nontrivial theorems



Future Direction: Tools for Lean Users

- We focus on enabling machine learning researchers to work on theorem proving
- LeanDojo is also useful for building practical proof automation tools for Lean users
- LLMStep
    - Work by Sean Welleck and Rahul Saha
    - Finetune LLMs for tactic generation on LeanDojo Benchmark
    - Integrate into Lean's VSCode workflow



Team

Aidan Swope

Alex Gu

Rahul Chalamala

Peiyang Song

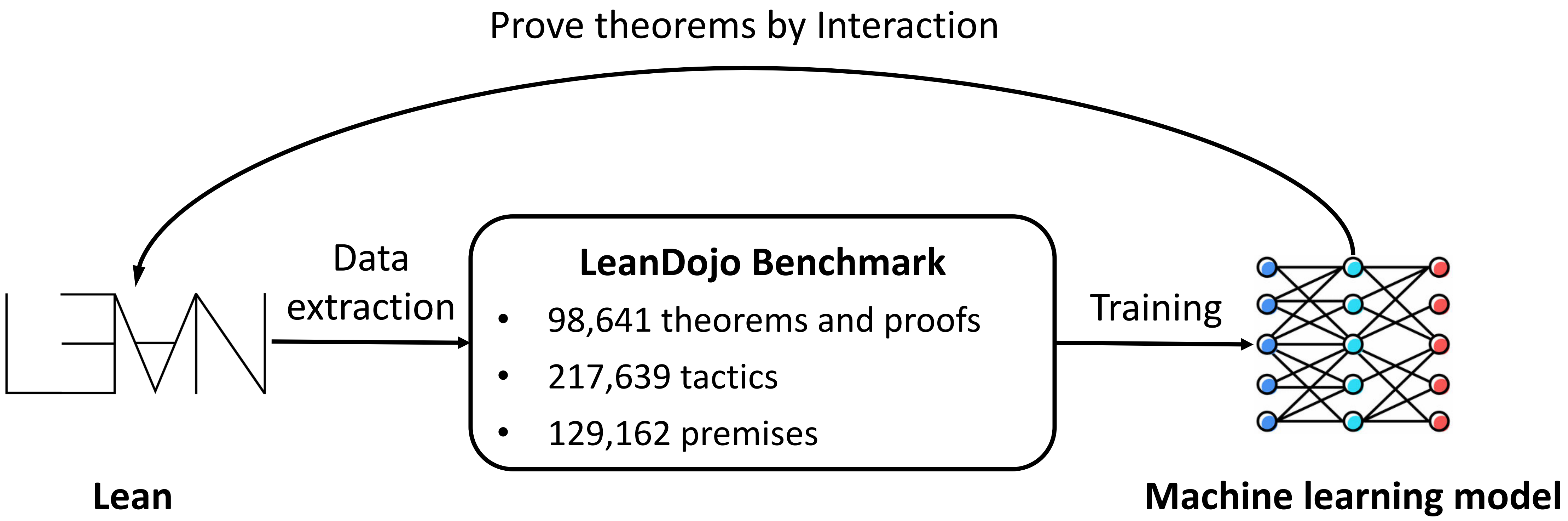
Shixing Yu

Saad Godil

Ryan Prenger

Anima Anandkumar

LeanDojo


Prove theorems by Interaction
Data extraction
LeanDojo Benchmark
• 98,641 theorems and proofs
• 217,639 tactics
• 129,162 premises
Training
Lean
Machine learning model