

LeanDojo: Theorem Proving with Retrieval-Augmented Language Models

Kaiyu Yang

Postdoc @ Computing + Mathematical Sciences



Caltech

Teaser: LLMs as Copilots for Theorem Proving

Formal Theorem Proving

Theorem

```
theorem set_inter_comm (s t : Set α) : s ∩ t = t ∩ s
```



Proof

```
ext x
simp [Set.mem_inter_iff]
constructor
· rintro {xs, xt}
| exact {xt, xs}
· rintro {xt, xs}
| exact {xs, xt}
```

Formal Theorem Proving

[Hales et al., "A Formal Proof of the Kepler Conjecture", 2017]
[Leroy et al., "CompCert - A Formally Verified Optimizing Compiler", 2016]

Theorem

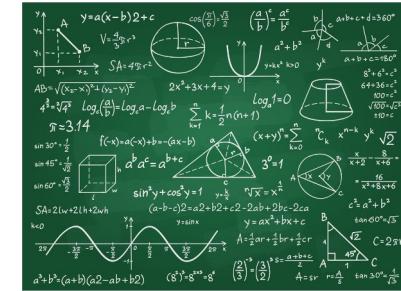
```
theorem set_inter_comm (s t : Set α) : s ∩ t = t ∩ s
```

```
ext x
simp [Set.mem_inter_iff]
constructor
· rintro {xs, xt}
| exact {xt, xs}
· rintro {xt, xs}
| exact {xs, xt}
```

Proof

- Theorems/proofs represented formally as programs

Formalize



Mathematics

Software

Formal Theorem Proving

[Hales et al., "A Formal Proof of the Kepler Conjecture", 2017]
[Leroy et al., "CompCert - A Formally Verified Optimizing Compiler", 2016]

Theorem

```
theorem set_inter_comm (s t : Set α) : s ∩ t = t ∩ s
```



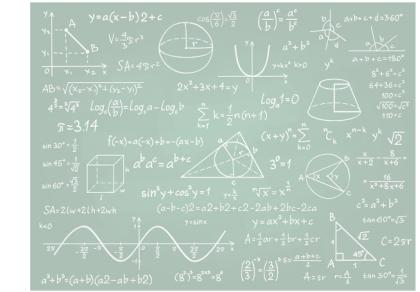
Proof

```
ext x
simp [Set.mem_inter_iff]
constructor
· rintro {xs, xt}
  exact {xt, xs}
· rintro {xt, xs}
  exact {xs, xt}
```



- Theorems/proofs represented formally as programs
- Proofs can be checked easily. No room for hallucination

Formalize



Mathematics

```
attachEvent('onreadystatechange',h).e.attachEvent('onload',function() {
  var t=0;
  if (document.readyState=="loading"||document.readyState=="uninitialized") return;
  var e=document.documentElement;
  if (e.attachEvent) e.attachEvent("onreadystatechange",h);
  else if (e.addEventListener) e.addEventListener("load",h,false);
  else if (e.createEventObject) e.createEventObject();
});
```

Software

Formal Theorem Proving

[Hales et al., "A Formal Proof of the Kepler Conjecture", 2017]
[Leroy et al., "CompCert - A Formally Verified Optimizing Compiler", 2016]

Theorem

```
theorem set_inter_comm (s t : Set α) : s ∩ t = t ∩ s
```



Proof

```
ext x
simp [Set.mem_inter_iff]
constructor
· rintro {xs, xt}
  exact {xt, xs}
· rintro {xt, xs}
  exact {xs, xt}
```



```
ext x
simp [Set.mem_inter_iff]
rintro {xs, xt}
exact {xt, xs}
```



Formalize

Mathematics

```
attachEvent('onreadystatechange',h),e.attachEvent('onload',function(){});function F(e){var t=e;return b.eventNames[t]==!1&&e.stopOnFalse?{r:!1,bubbles:!0}:{};var o=u.length:r&&(s+t,c(r));return this,removeEventListener(t,o,u,!0);};var p=o.createEvent('Event');p.initEvent('click',!0,!0);var n=h.call(arguments),t=n.length,i=1==r||e&e(r),l=Array(r);for(t;i<=r;i++)n[t]&&b.isFunction(n[t])&&(l[i]=n[t]());for(i=0;i<t;i++)l[i].call(this);for(i=t;i<n.length;i++)n[i].call(this);var r=n[n.length-1];r.value=p,r.style.cssText='border:1px solid black';var s=r.getAttribute('style');r.setAttribute('style',s);r.click();};
```

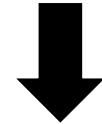
Software

- Theorems/proofs represented formally as programs
- Proofs can be checked easily. No room for hallucination

How to Prove Theorems (with Machine Learning)?

Proof Assistants (Interactive Theorem Provers)

Theorem

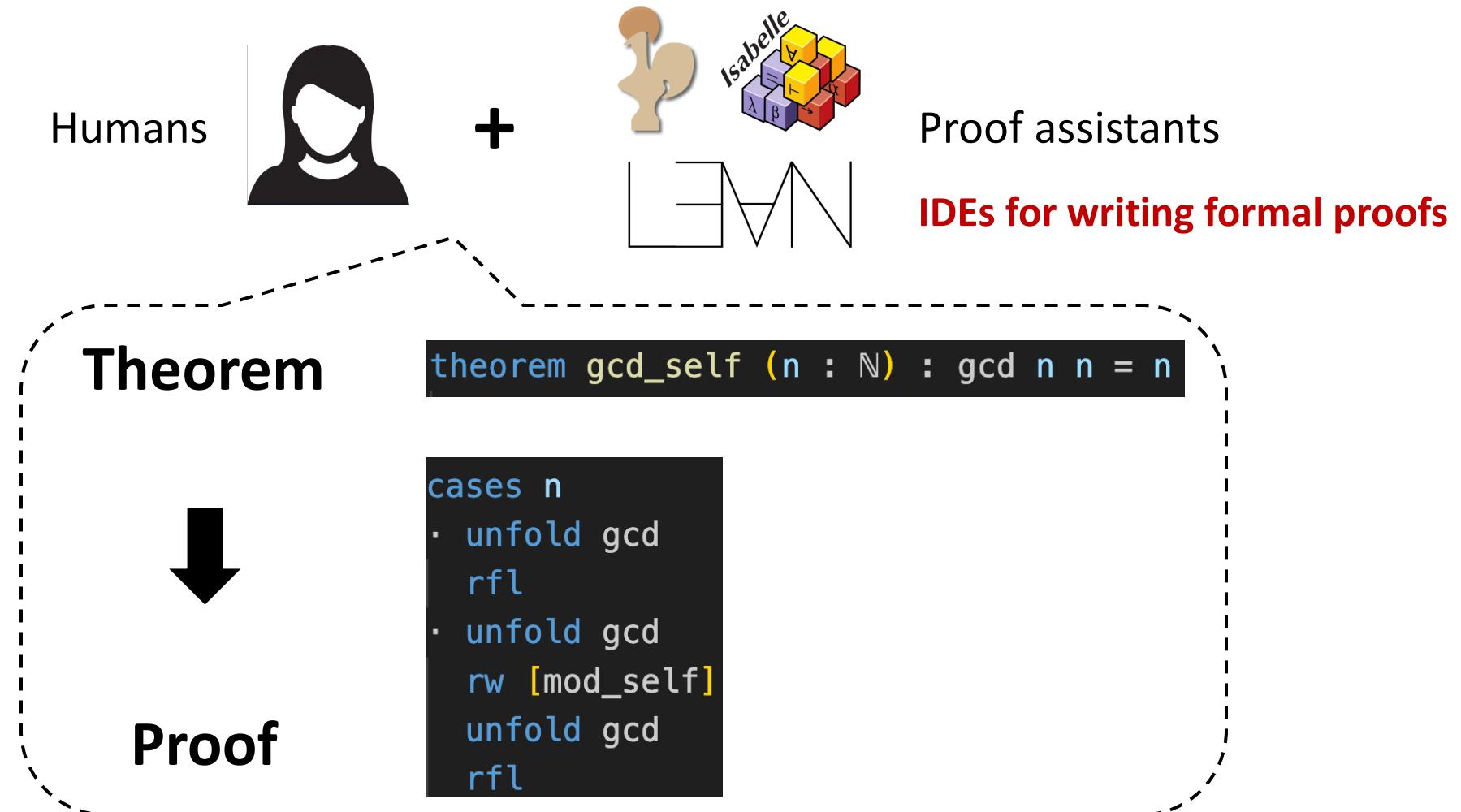


Proof

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

```
cases n
· unfold gcd
  rfl
· unfold gcd
  rw [mod_self]
  unfold gcd
  rfl
```

Proof Assistants (Interactive Theorem Provers)



Generating Proof Steps (Tactics)

```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
  intro a b c
  rw [Nat.add_right_comm]
```

Generating Proof Steps (Tactics)

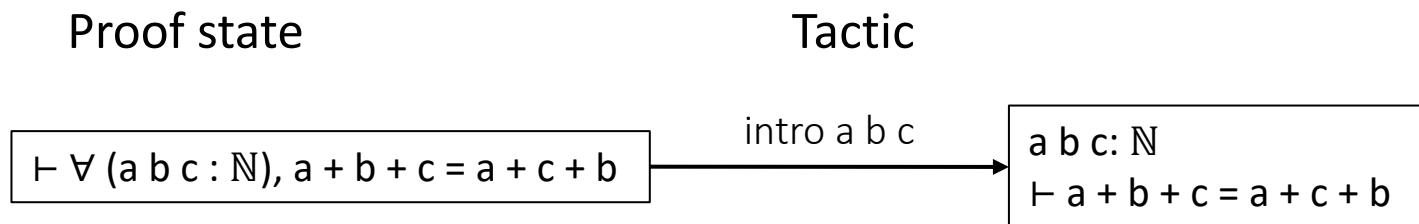
```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
  intro a b c
  rw [Nat.add_right_comm]
```

Proof state

```
↑ ∀ (a b c : ℕ), a + b + c = a + c + b
```

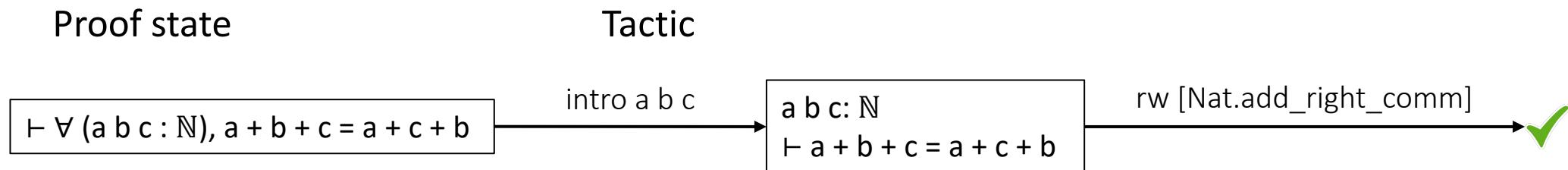
Generating Proof Steps (Tactics)

```
theorem add_abc : ∀ a b c : ℙ, a + b + c = a + c + b := by
  intro a b c
  rw [Nat.add_right_comm]
```



Generating Proof Steps (Tactics)

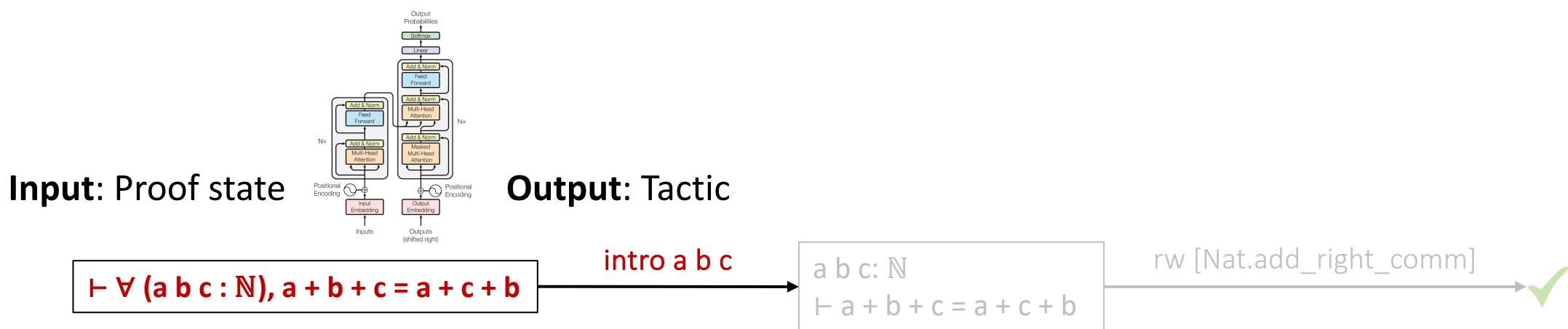
```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
  intro a b c
  rw [Nat.add_right_comm]
```



Generating Proof Steps (Tactics)

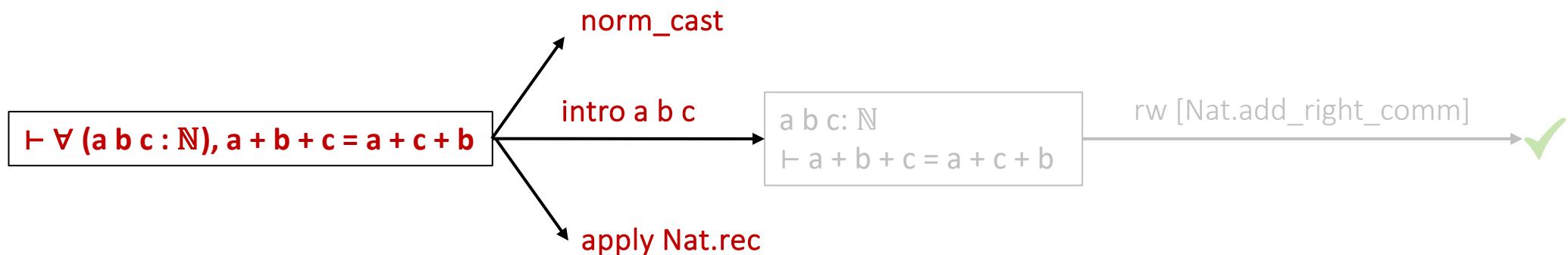
```
theorem add_abc : ∀ a b c : ℙ, a + b + c = a + c + b := by
  intro a b c
  rw [Nat.add_right_comm]
```

Tactic generator



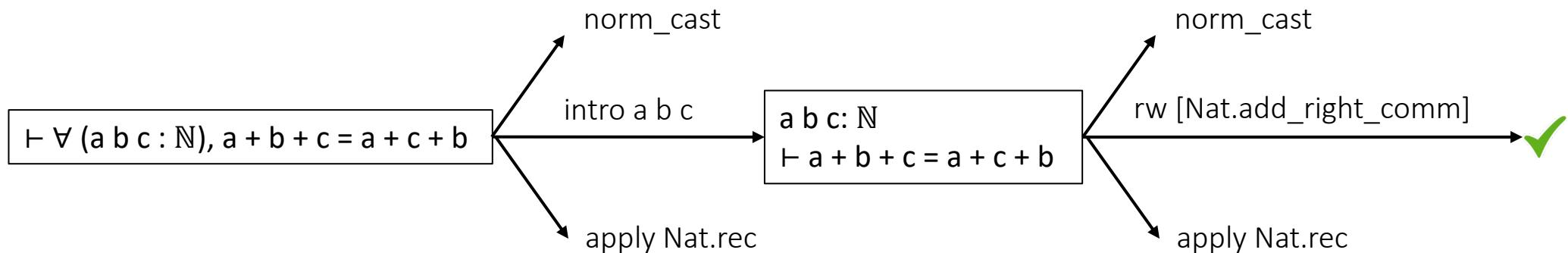
Generating Proof Steps (Tactics)

```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
| intro a b c
| rw [Nat.add_right_comm]
```



Searching for Proofs

```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
| intro a b c
| rw [Nat.add_right_comm]
```

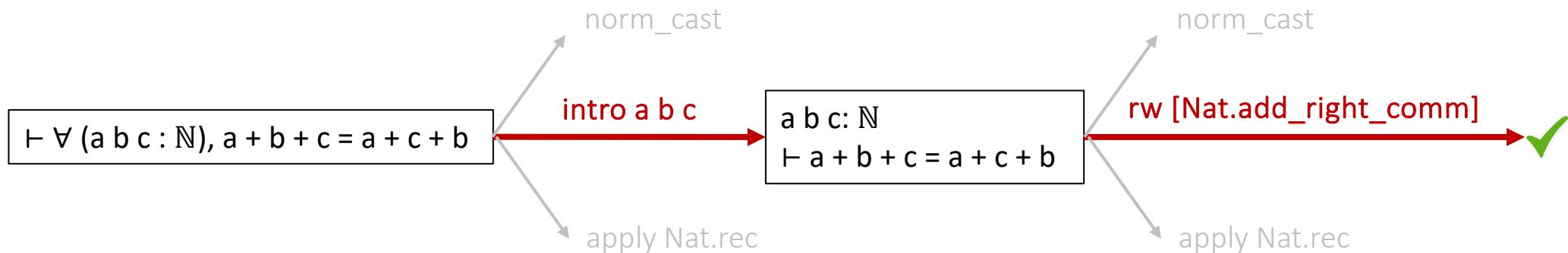


Searching for Proofs

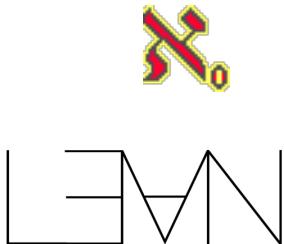
```
theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
| intro a b c
| rw [Nat.add_right_comm]
```

Classical proof search algorithms

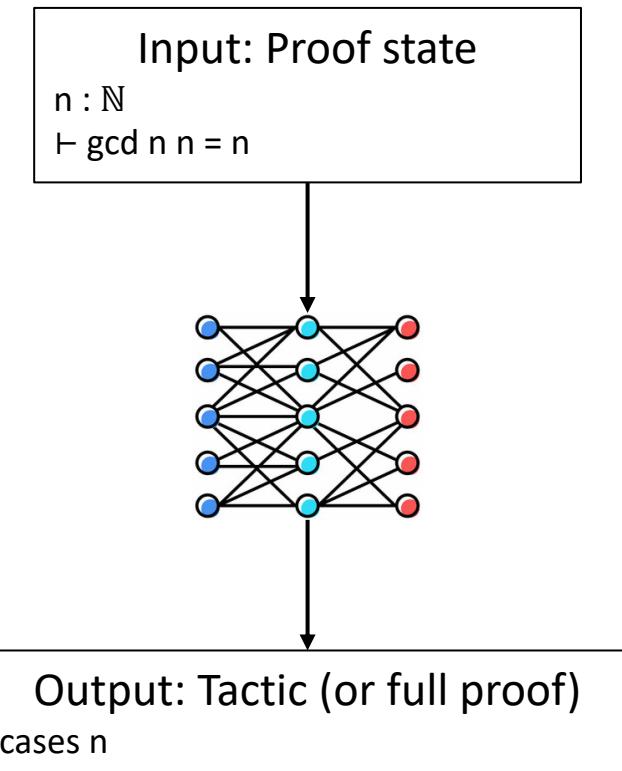
- Depth first search (DFS)
- Breadth first search (BFS)
- ...



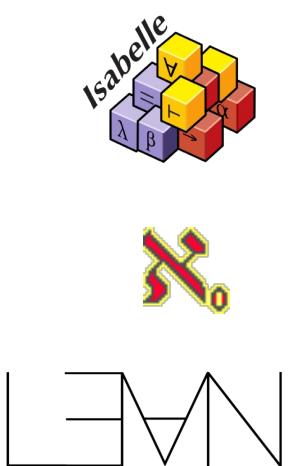
LLMs for Theorem Proving



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

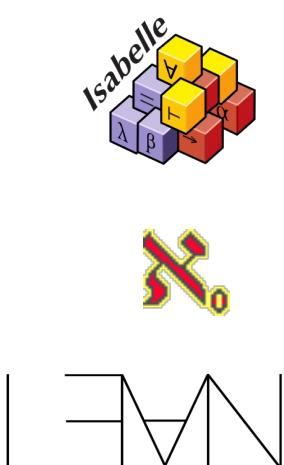


LLMs for Theorem Proving



	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

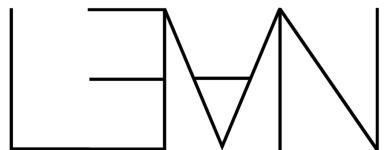
LLMs for Theorem Proving



	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

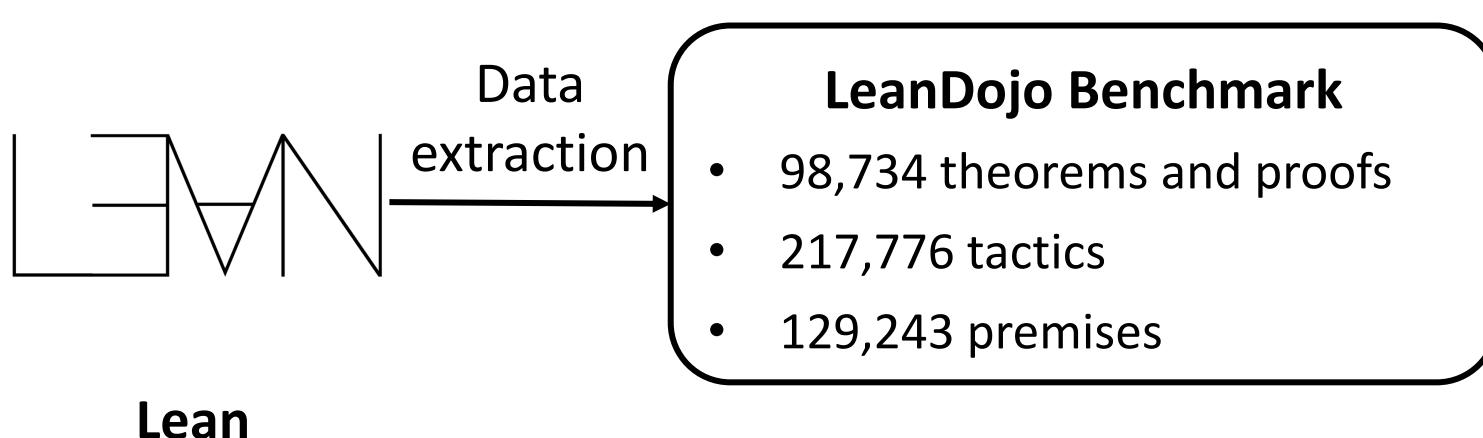
[Yang et al., "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models", NeurIPS 2023]

LeanDojo

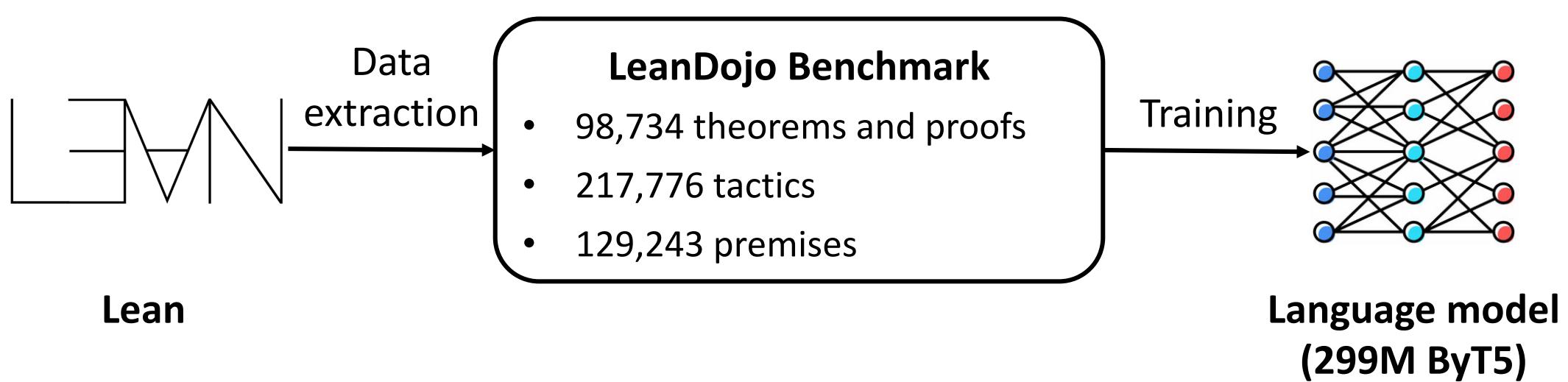


Lean

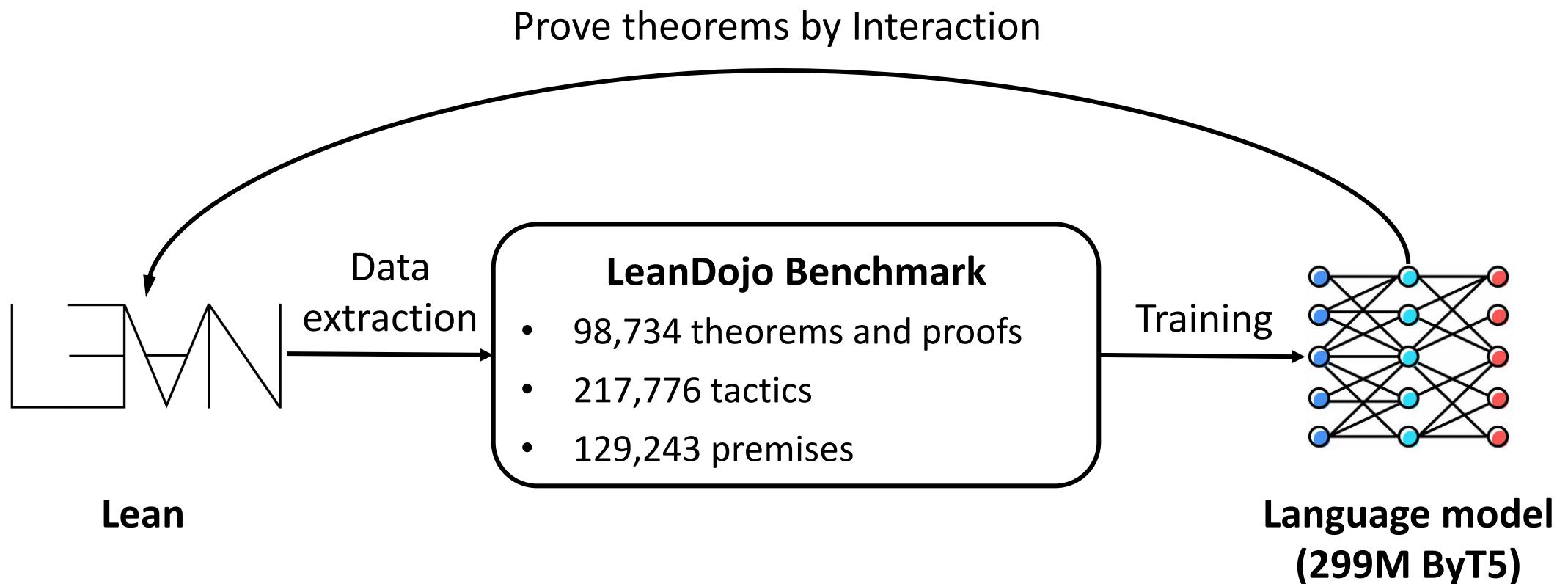
LeanDojo



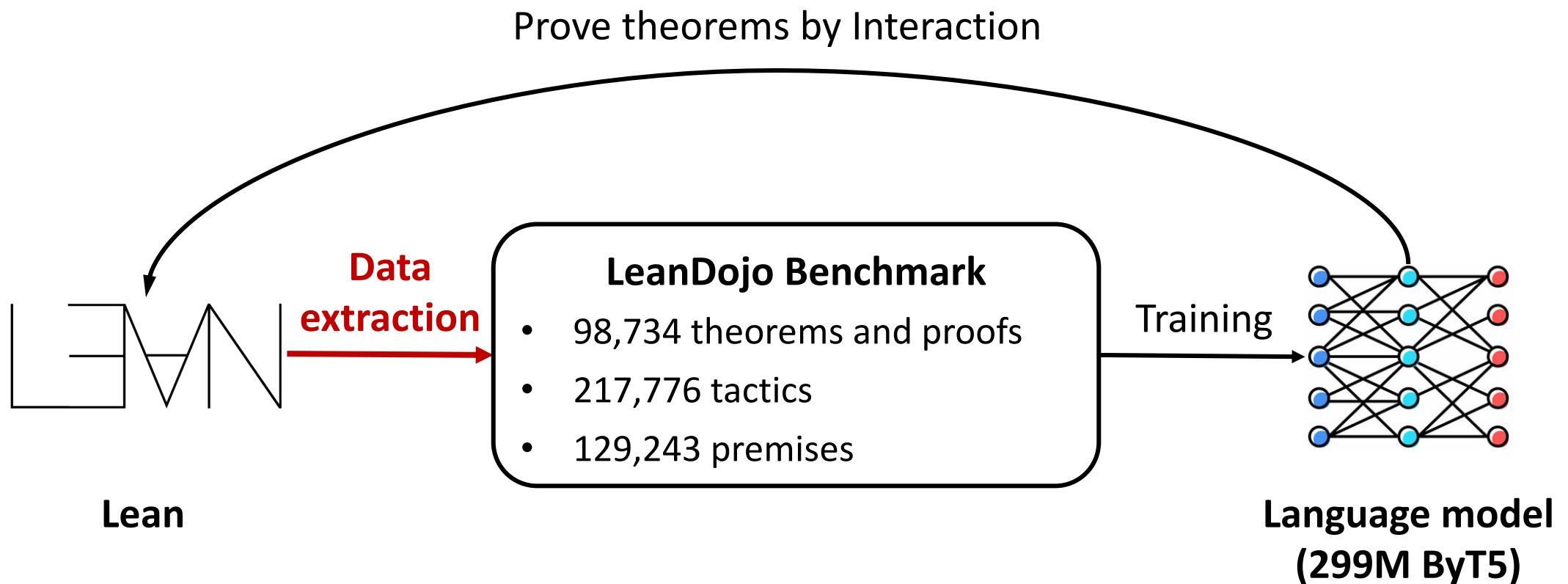
LeanDojo



LeanDojo



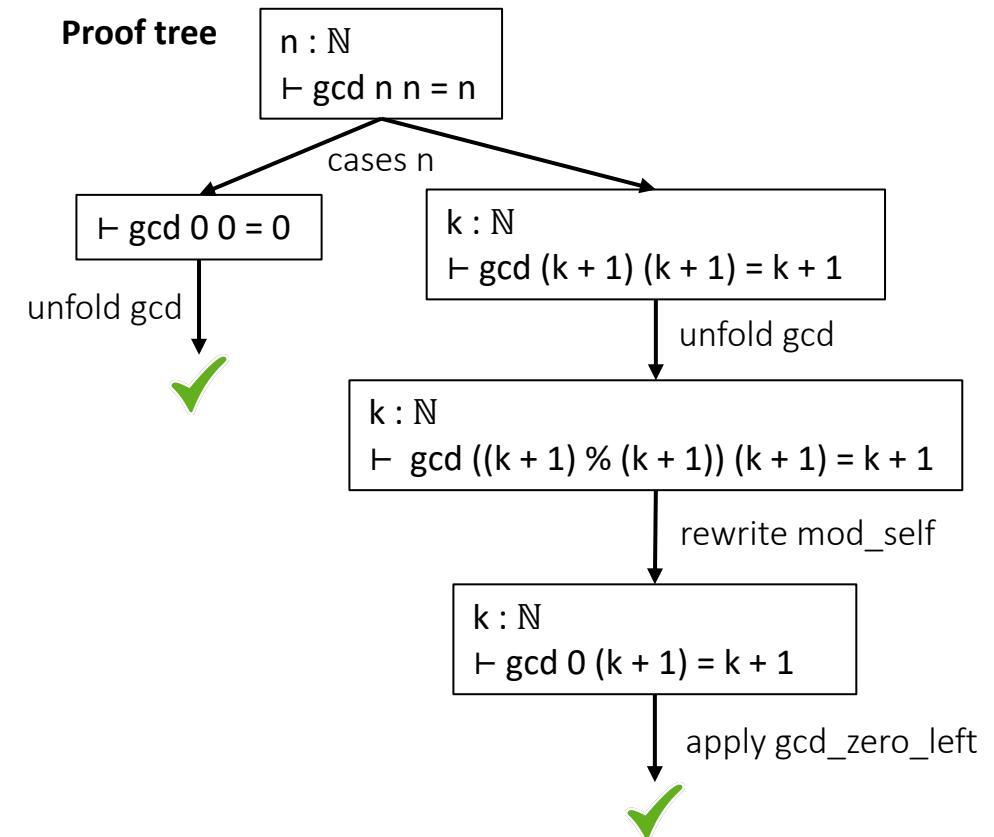
LeanDojo



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

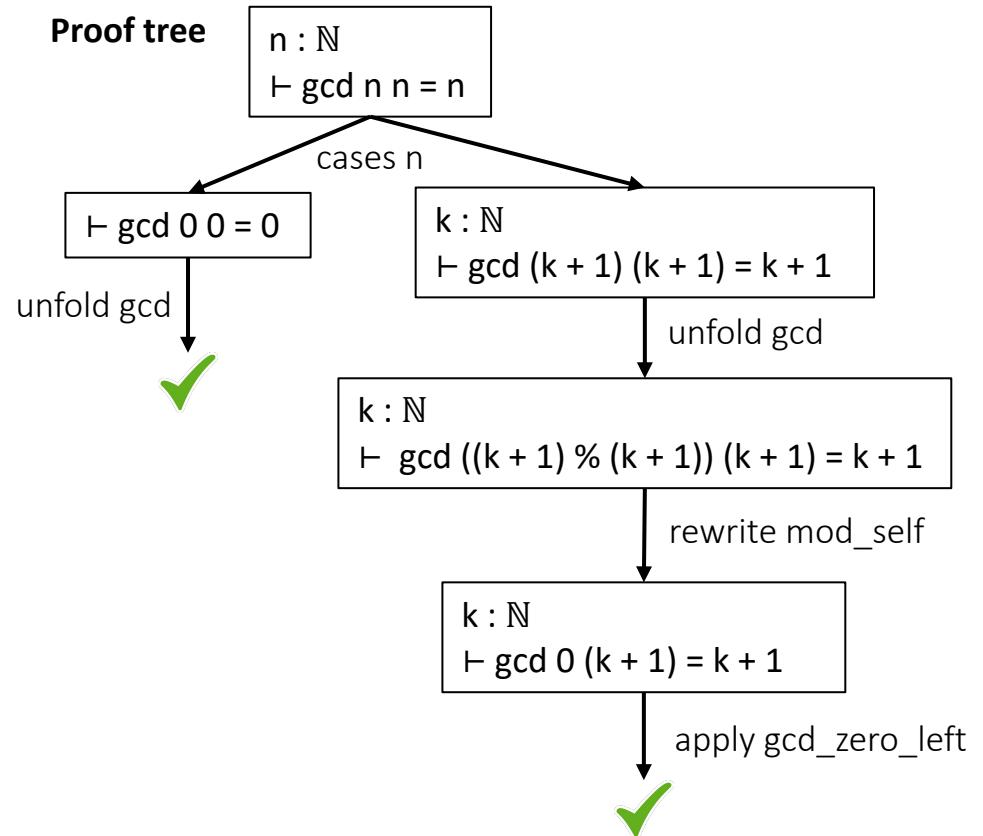
```
theorem gcd_self (n : nat) : gcd n n = n :=
begin
  cases n,
  { unfold gcd },
  unfold gcd,
  rewrite mod_self,
  apply gcd_zero_left
end
```



Extracting Premises

- 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

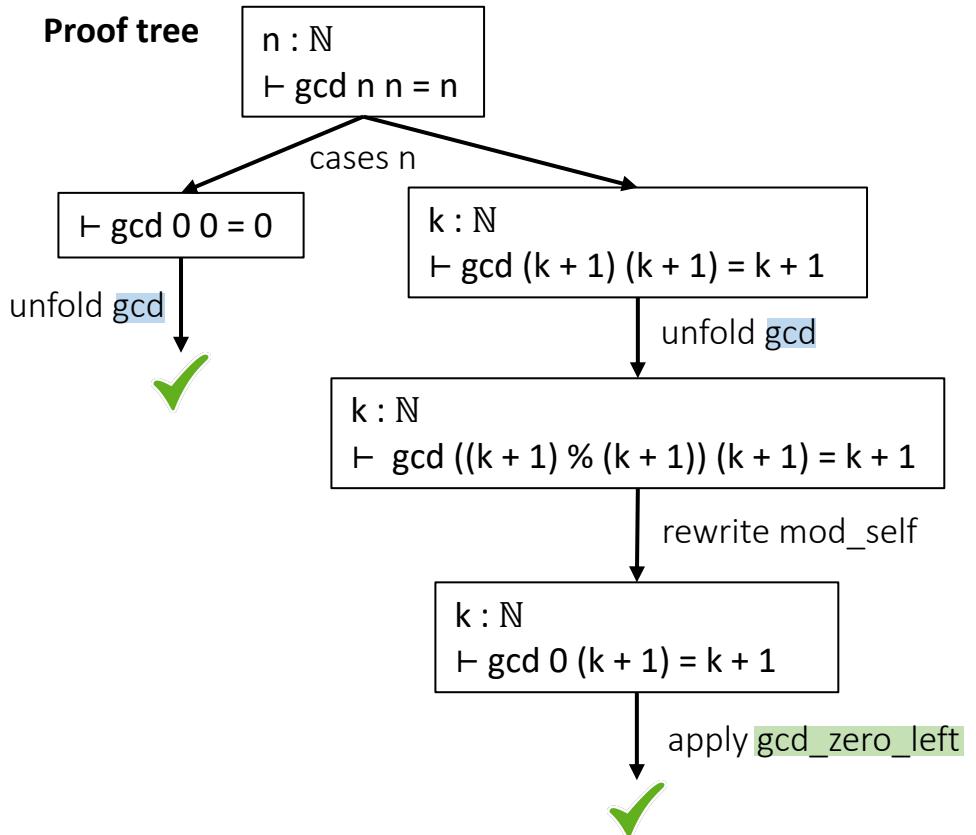
- Tactics rely on **premises**
 - Lemmas
 - Definitions

data/nat/gcd.lean

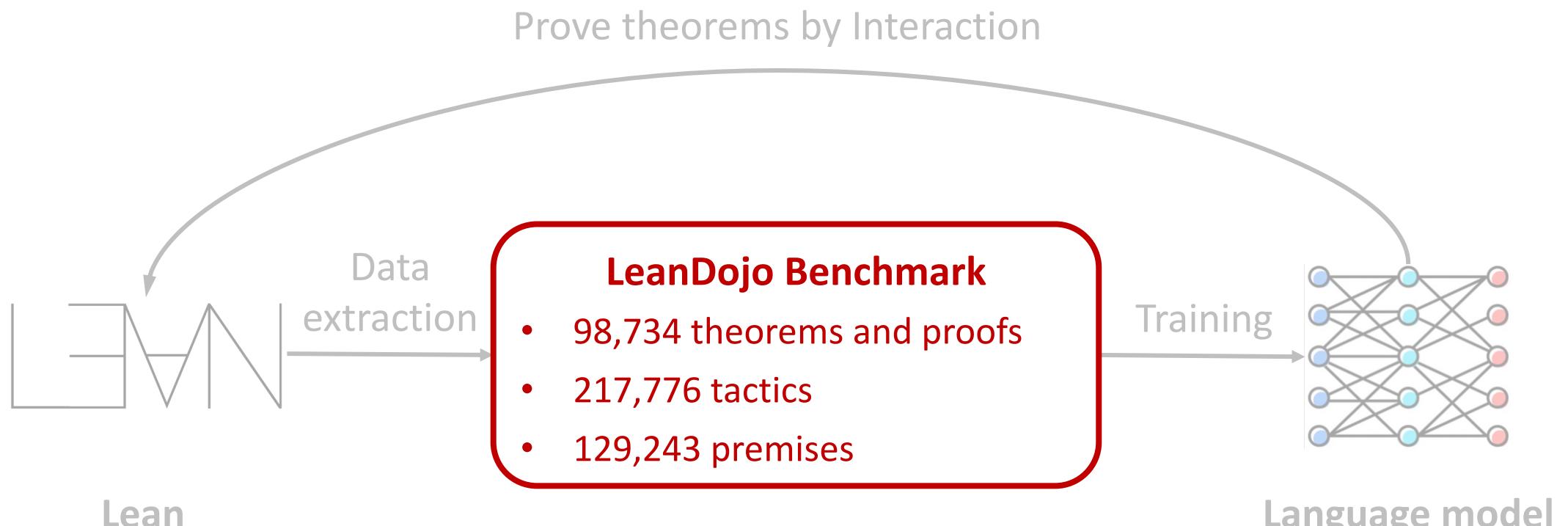
```
def gcd : nat → nat → nat          -- gcd z y
| 0      y := y                  -- Case 1: z == 0
| (x + 1) y := gcd (y % (x + 1)) (x + 1)  -- Case 2: z > 0

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
```



LeanDojo



From mathlib: <https://github.com/leanprover-community/mathlib>

Easy to construct your own benchmarks

Challenging Data Split

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

Challenging Data Split

- random: Splitting theorems into training/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

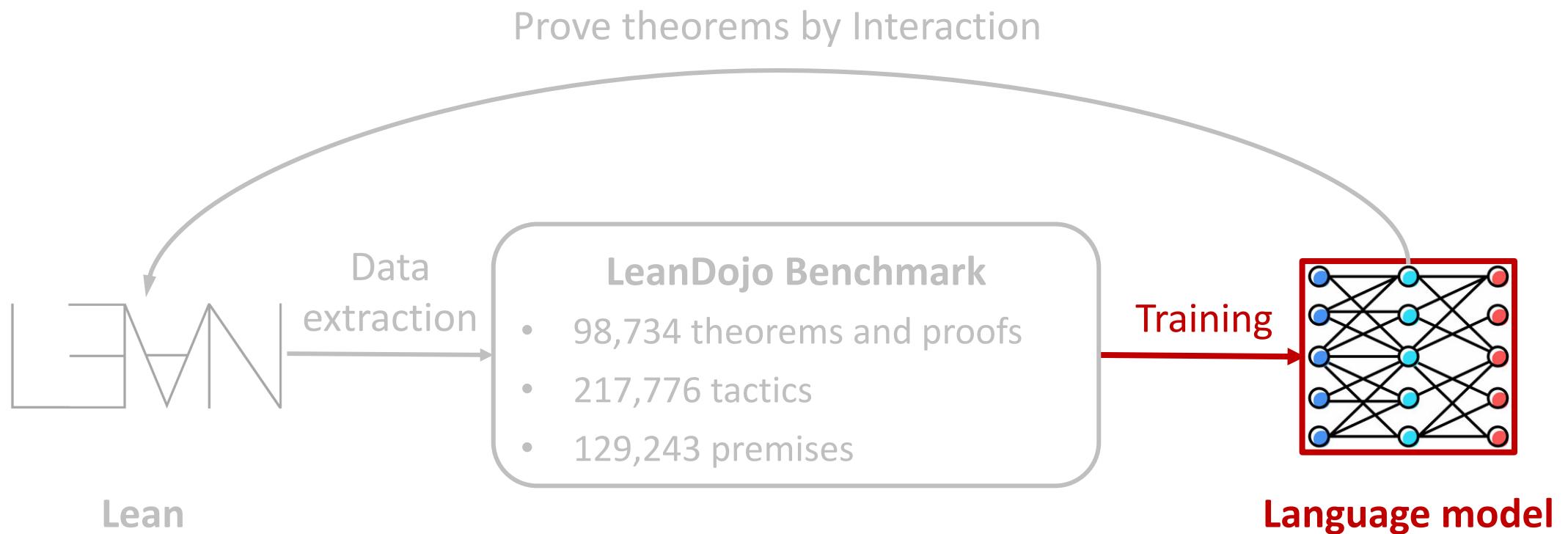
```
lemma conj_conj_mul : (a.conj * b).conj = b.conj * a := begin
  rw [conj_mul, conj_conj]
end
```

Test

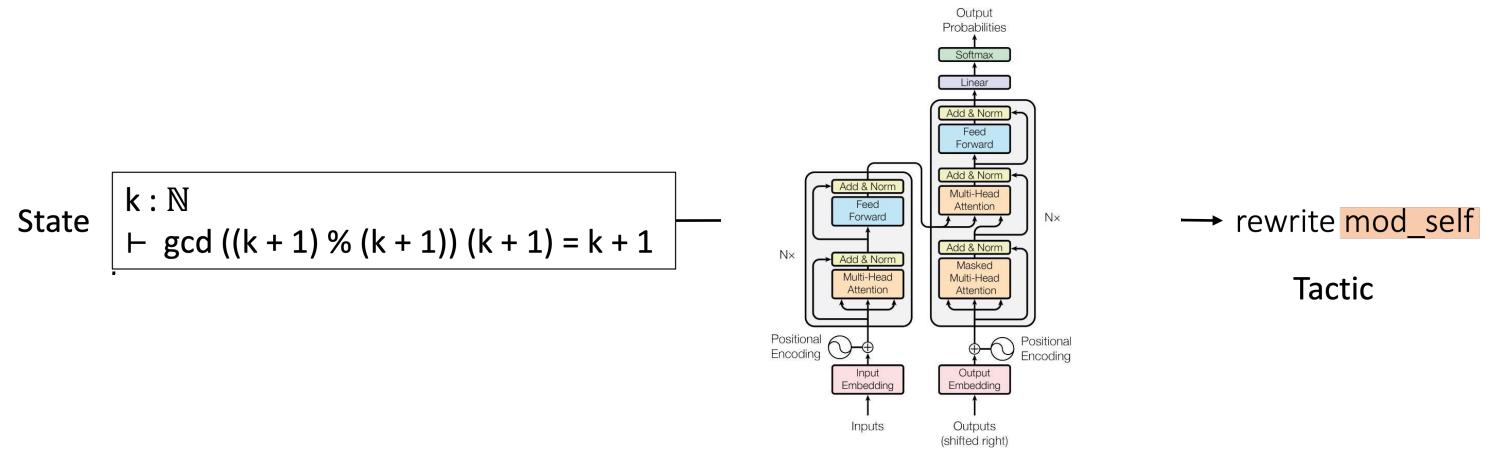
```
lemma conj_mul_conj : (a * b.conj).conj = b * a.conj := begin
  rw [conj_mul, conj_conj]
end
```

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

LeanDojo



Existing Models for Tactic Generation



ReProver: Retrieval-Augmented Prover

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

All *accessible premises*
in the math library

State
$$\boxed{k : \mathbb{N} \\ \vdash \text{gcd } ((k + 1) \% (k + 1)) (k + 1) = k + 1}$$

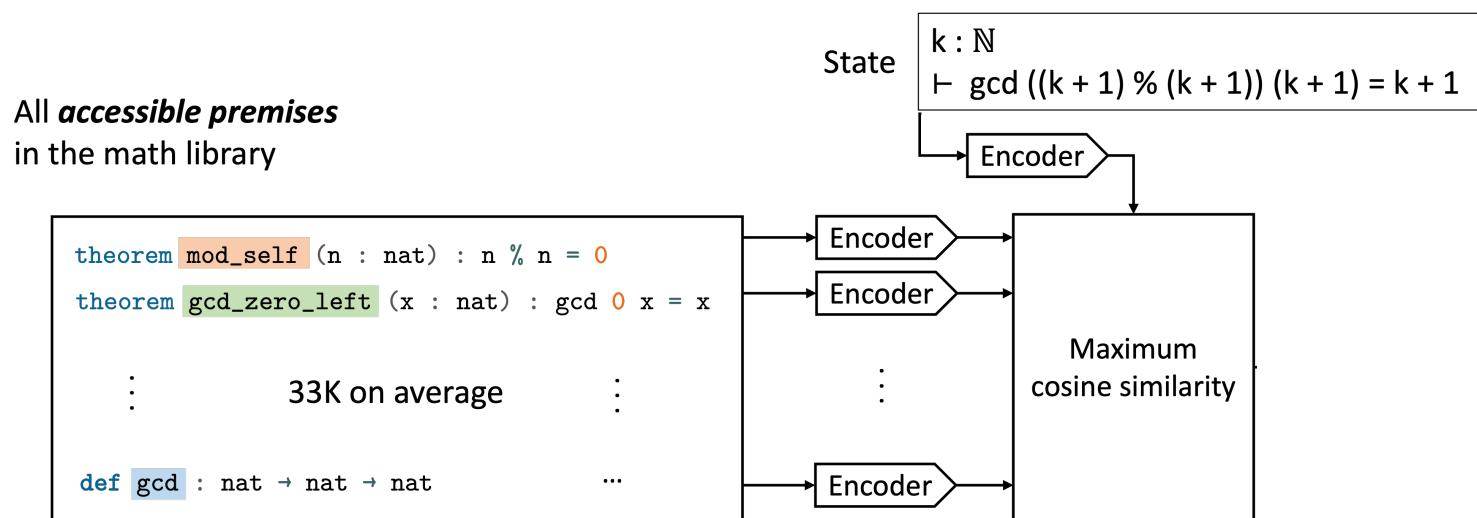
```
theorem mod_self (n : nat) : n \% n = 0
theorem gcd_zero_left (x : nat) : gcd 0 x = x

:
  33K on average
:

def gcd : nat → nat → nat
...
```

ReProver: Retrieval-Augmented Prover

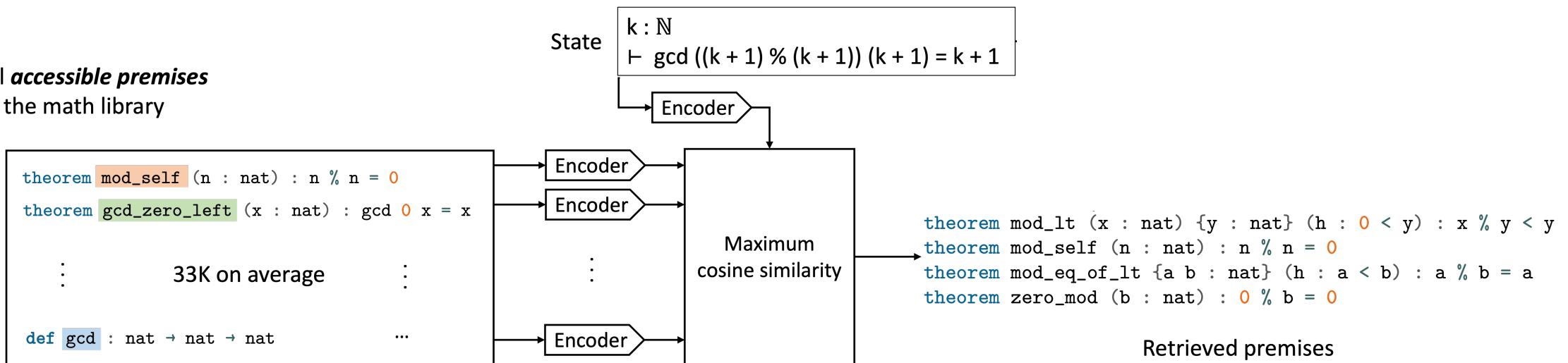
- Given a state, we retrieve premises from the set of **all accessible premises**



ReProver: Retrieval-Augmented Prover

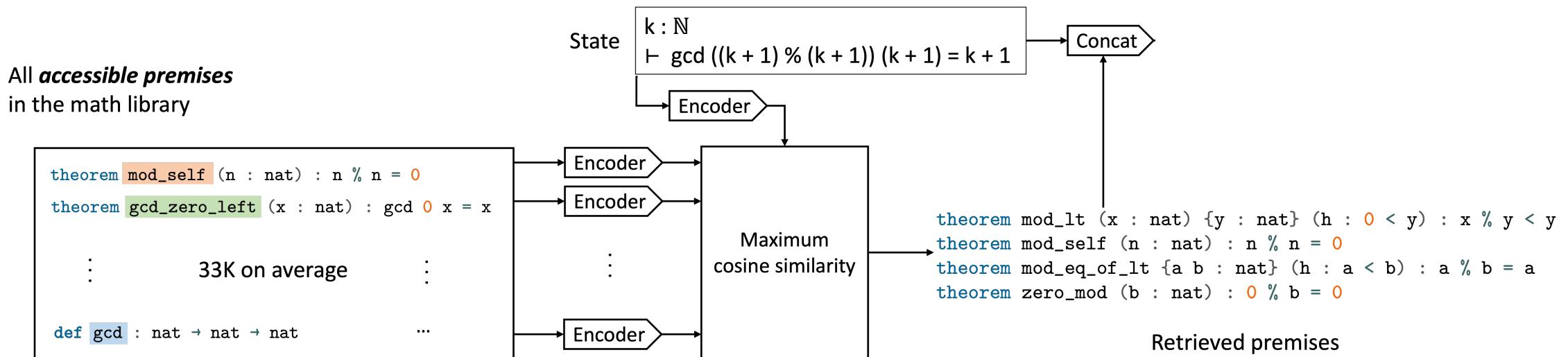
- Given a state, we retrieve premises from the set of **all accessible premises**

All *accessible premises*
in the math library



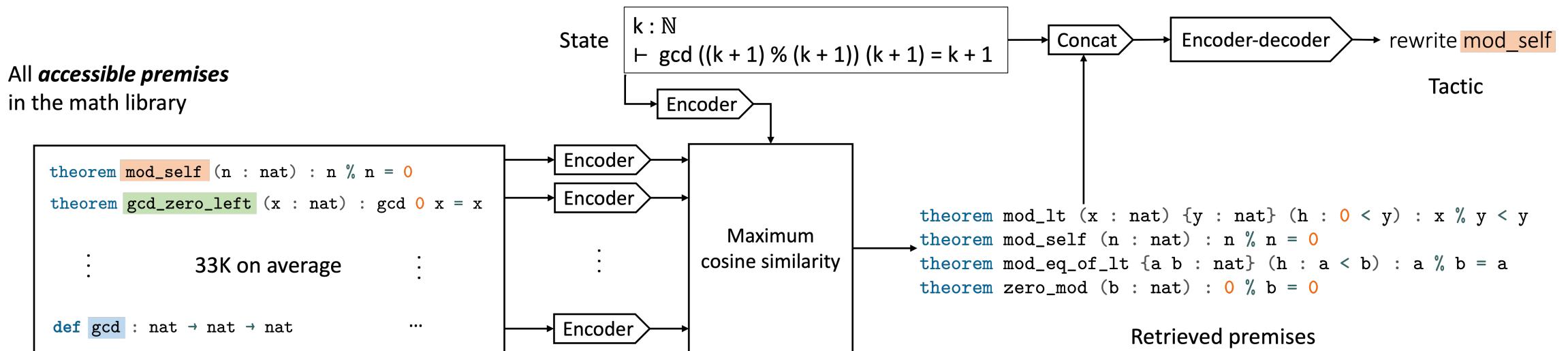
ReProver: Retrieval-Augmented Prover

- 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

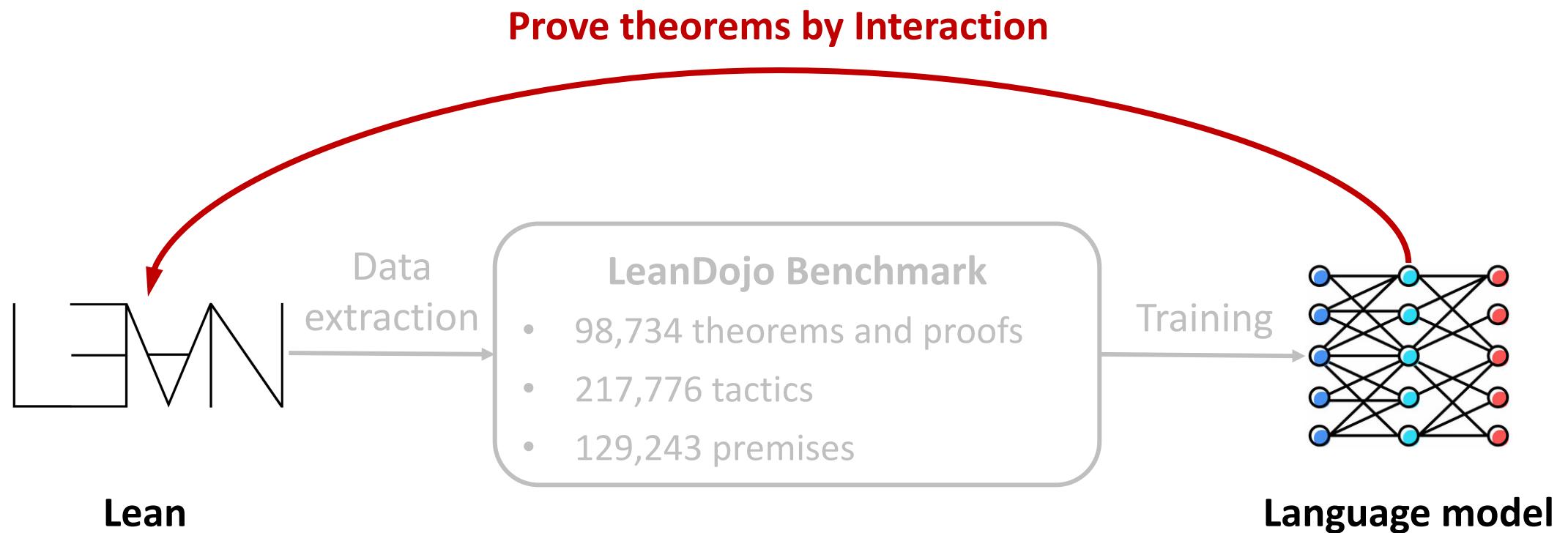


ReProver: Retrieval-Augmented Prover

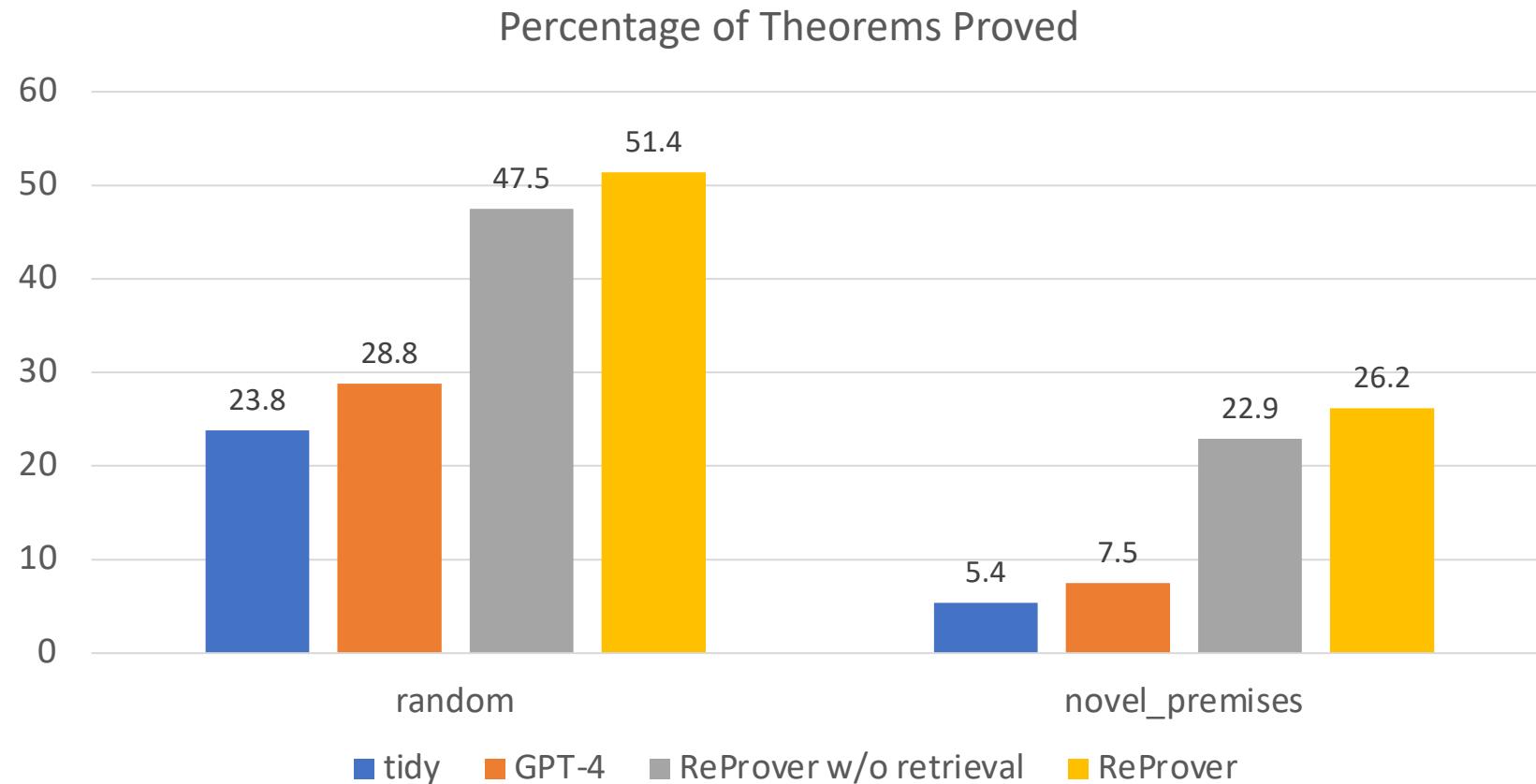
- Given a state, we retrieve premises from the set of **all accessible premises**
- Retrieved premises are concatenated with the state and used for tactic generation



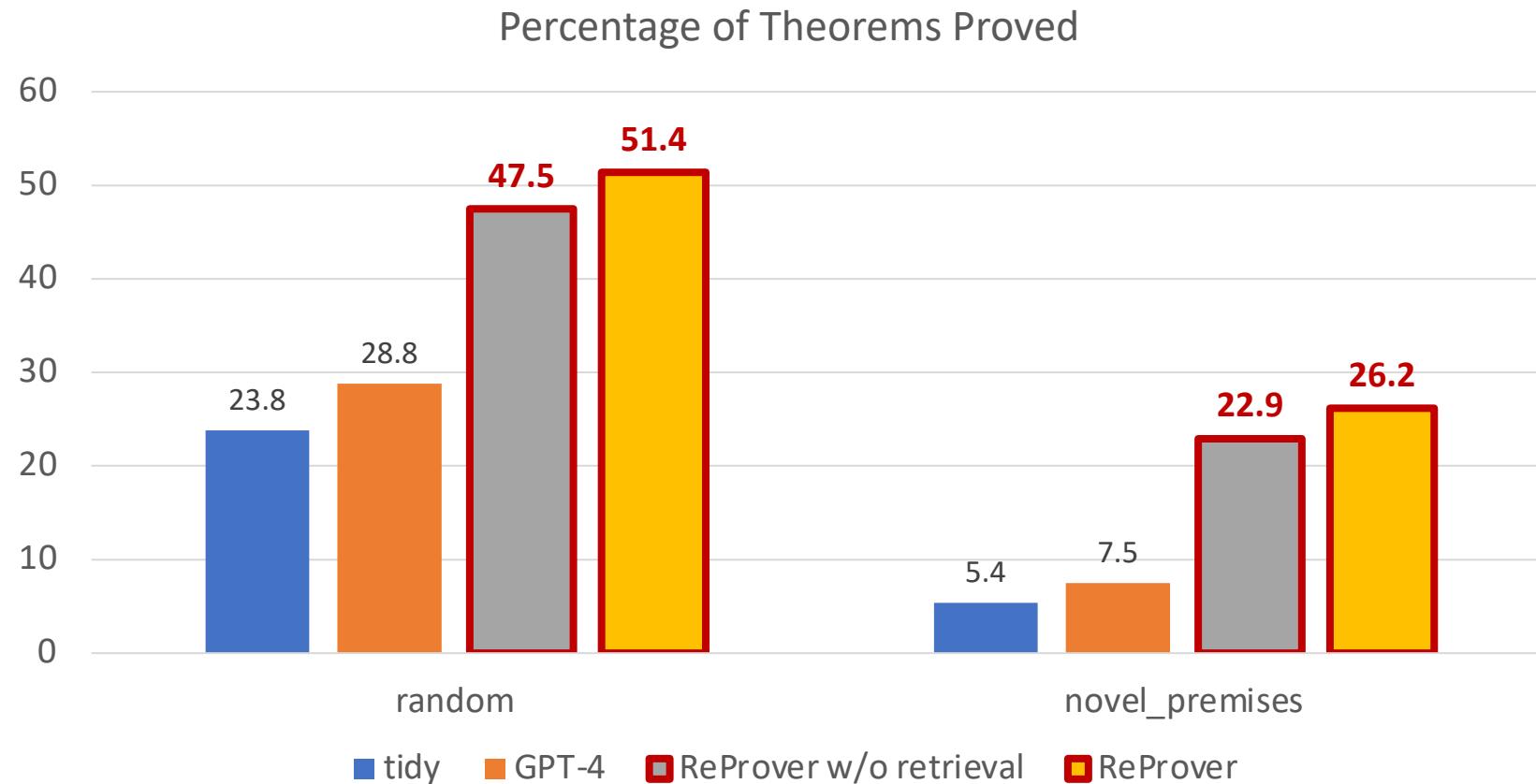
LeanDojo



Premise Retrieval Improves Theorem Proving



Premise Retrieval Improves Theorem Proving



Results on MiniF2F

- MiniF2F: Benchmark for formal math Olympiads [Zheng et al., MiniF2F, ICLR 2022]
- ReProver is competitive to state of the art
 - 26.5% vs. 25.9% (w/o RL) [Polu et al., "Formal Mathematics Statement Curriculum Learning", ICLR 2023]
 - Smaller model: 517M vs. 774M
 - Much smaller compute: 120 hours vs. 48K hours (We don't pretrain on math-specific data)
 - Open-source!

Discovering New Proofs ON ProofNet

[Azerbayev et al., ProofNet, 2023]

We evaluate the model on 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⁻¹ (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
```

LLMs as Copilots for Theorem Proving

The image shows a code editor interface with two panels. The left panel displays the file `Lean4Example.lean` containing Lean code. The right panel shows the `Lean Infoview`, which provides information about the current tactic state and suggests tactics for the goal.

`Lean4Example.lean` content:

```
1 import LeanInfer
2
3 open Nat (add_assoc add_comm)
4
5 def hello := "world"
6
7 theorem hello_world (a b c : Nat)
8   : a + b + c = a + c + b := by
9   rw [add_assoc, add_comm b, ←add_assoc]
10
11 theorem foo (a : Nat) : a + 1 = Nat.succ a := by
12   suggest_tactics
13   rfl
14
```

`Lean Infoview` content:

- ▼ Lean4Example.lean:12:2
- ▼ Tactic state
 - 1 goal
 - a** : Nat
 - $\vdash a + 1 = \text{Nat.succ } a$
- ▼ Tactic suggestions
 - Try this:
 - `induction a with`
 - | zero => rfl
 - | succ => rfl
 - `simp only [Nat.succ_sub_one]`
 - `cases a <;> rfl`
 - `rfl`
 - `simp`
 - `cases a`

[Song et al., "Towards Large Language Models as Copilots for Theorem Proving in Lean", NeurIPS MATH-AI Workshop 2023]

Posters

LeanDojo: 10:45 AM TODAY!

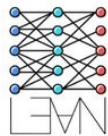
Lean Copilot: 3 PM Friday @ MATH-AI Workshop

Open Code, Models, Data

lean-dojo/ LeanDojo

Tool for data extraction and interacting with Lean programmatically.

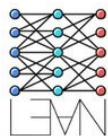
7 Contributors 2 Used by 13 Discussions 344 Stars 41 Forks



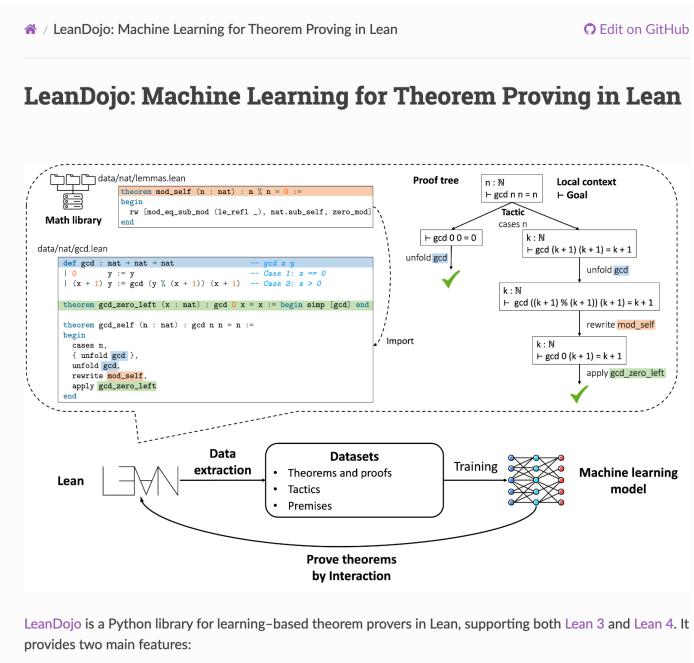
lean-dojo/ LeanCopilot

Native Neural Network Inference in Lean

5 Contributors 3 Issues 1 Discussion 67 Stars 4 Forks



The screenshot shows the GitHub repository page for LeanDojo. It includes the repository name, a brief description, navigation links like 'Getting Started' and 'User Guide', and a sidebar with developer statistics. An advertisement for EthicalAds is displayed at the bottom.



The screenshot shows the Hugging Face page for the 'kaiyuy/leandojo-lean4-tacgen-byt5-small' model. It includes the model name, a smiling emoji, a download count of 446 last month, and a purple line graph representing the download trend. Below the graph, there are buttons for 'Text2Text Generation', 'Transformers', 'PyTorch', 't5', 'Inference Endpoints', and 'text-generation-inference', along with a 'License: mit' link.

Team



Aidan Swope



Alex Gu



Rahul Chalamala



Peiyang Song



Shixing Yu



Saad Godil

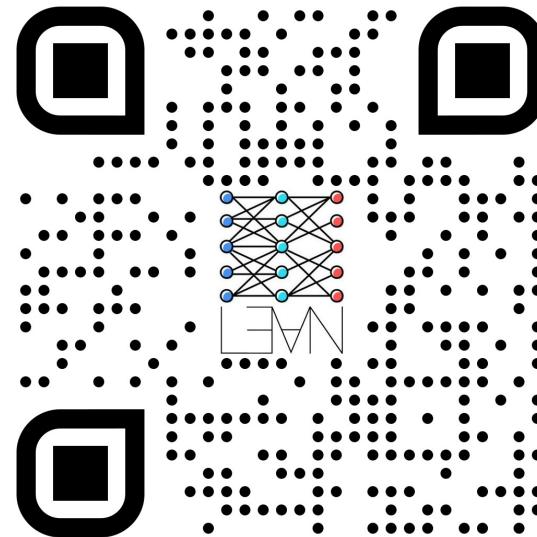


Ryan Prenger



Anima Anandkumar

LeanDojo



- Project webpage: leandojo.org
 - **Poster: Today 10:45 AM–12:45 PM**

