Proof Artifact Co-Training for Theorem Proving with Language Models

Jesse Michael Han
University of Pittsburgh

Jason Rute
CIBO Technologies

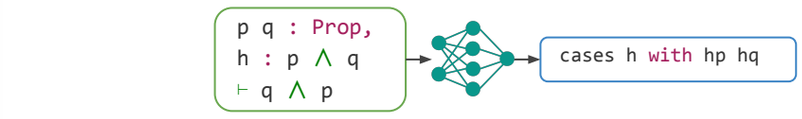
Yuhuai Wu
University of Toronto

Edward Ayers
University of Cambridge

Stanislas Polu
OpenAI

Labeled data for imitation learning of theorem proving in large libraries of formalized mathematics is scarce, as such libraries require years of concentrated effort by human specialists to be built. This is particularly challenging when applying large Transformer language models to tactic prediction, because the scaling of performance with respect to model size is quickly disrupted in the data-scarce, easily-overfitted regime. We propose **PACT** (Proof Artifact Co-Training), a general methodology for extracting abundant self-supervised data from kernel-level proof terms for co-training alongside the usual tactic prediction objective. We apply this methodology to **Lean**, an interactive proof assistant which hosts some of the most sophisticated formalized mathematics to date. We instrument Lean with a neural theorem prover driven by a Transformer language modeland show that PACT improves theorem proving success rate on a held-out suite of test theorems from **32%** to **48%.**

Applying language modelling to theorem proving



Training Example:

```
GOAL p q : Prop, ⇥ h : p ∧ q ⇥ ⊢ q ∧ p PROOFSTEP cases h with hp hq
```

Inference Example:

```
GOAL a b : Prop, ⇥ h : a ∧ b ⇥ ⊢ b ∧ a PROOFSTEP
```

apply	0.10
cases	0.81
rcases	0.04

Repeatedly sample next token from distribution.

```
GOAL a b : Prop, ⇥ h : a ∧ b ⇥ ⊢ b ∧ a PROOFSTEP cases h with ha hb
```

cases	h	with	H a	h b

PACT: self-supervised data from proof terms

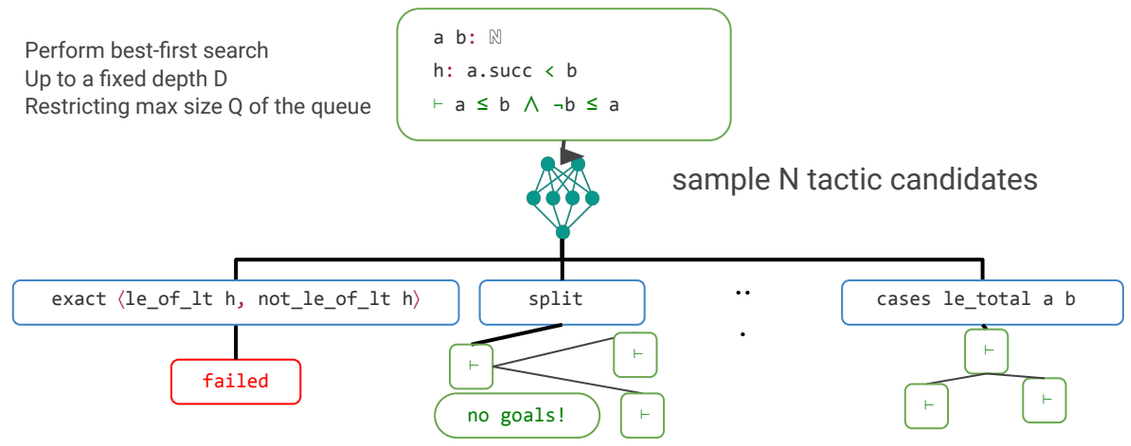
```
lemma iff_of_true {P : Prop}
  : (P ↔ true) → P :=
begin
  intro H,
  cases H with H₁ H₂,
  apply H₂,
  trivial
end
```

Masked proof term	Tactic state	Term that fulfils goal
_	⊢ (P ↔ true) → P	λ H, iff.dcases_on H (λ H₁ H₂, H₂ trivial)
λ H, _	H : P ↔ true ⊢ P	iff.dcases_on H (λ H₁ H₂, H₂ trivial)
λ H, iff.dcases_on H (λ H₁ H₂, _)	H₁ : P → true H₂ : true → P ⊢ P	H₂ trivial
λ H, iff.dcases_on H (λ H₁ H₂, H₂ _)	H₁ : P → true H₂ : true → P ⊢ true	trivial
λ H, _ H (λ H₁ H₂, H₂ trivial)	H : P ↔ true ⊢ (P ↔ true) → ((P → true) → (true → P) → P) → P	iff.dcases_on

```
GOAL P : Prop ⊢ (P ↔ true) → P PROOFSTEP intro H
GOAL P : Prop, H : P ↔ true ⊢ P PROOFSTEP cases H with H₁ H₂
GOAL P : Prop, H₁ : P → true, H₂ : true → P ⊢ P PROOFSTEP apply H₂
GOAL P : Prop, H₁ : P → true, H₂ : true → P ⊢ true PROOFSTEP trivial
...
RESULT λ {P : Prop} (H : P ↔ true), PREDICT H (λ (H₁ : P → true) (H₂ : true → P), H₂ trivial) TYPE (P ↔ true) → ((P → true) → (true → P) → P) → P
TYPE ∀ {P : Prop}, (P ↔ true) → P NAME iff_of_tlean
GOAL P : Prop, H : P ↔ true ⊢ (P ↔ true) → ((P → true) → (true → P) → P) → P NEXT_LEMMA iff.dcases_on
RESULT λ {P : Prop} (H : P ↔ true), PREDICT H (λ (H₁ : P → true) (H₂ : true → P), H₂ trivial) TYPE (P ↔ true) → ((P → true) → (true → P) → P) → P
```

Wrap language model with a best-first proof search

Perform best-first search
Up to a fixed depth D
Restricting max size Q of the queue



Why Lean?

Popular and newsworthy

Quanta magazine

Easy to learn and use

Community

Active and supportive community

GitHub

Extensive and growing math library

Con(ZFC - CH)

Perfectoid Spaces

Great tools and customization

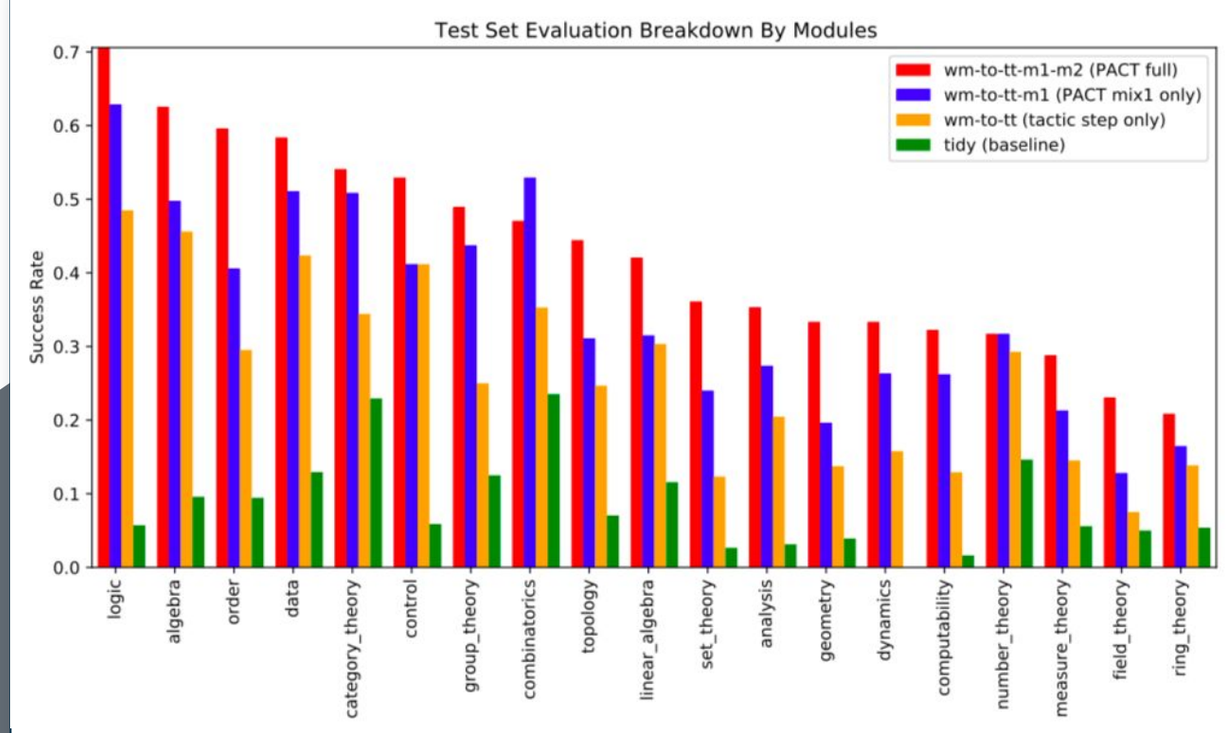
meta

Community-driven grand challenges

$$\mathrm{Ext}^i_{\mathrm{Cond(Ab)}}(\mathcal{M}_{p'}(S), V) = 0$$

Model	Tokens total	Early-stop	mix1	mix2	tactic	Pass-rate
Baselines						
refl						1.1%
tidy-bfs						9.9%
WebMath > tactic	32B	1B			1.02	32.2%
Pre-training						
WebMath > mix1	32B	11B	*0.08*			
WebMath > mix2	32B	16B		*0.08*		
WebMath > mix1 + mix2	32B	22B	*0.11*	*0.08*		
WebMath > mix1 > tactic	32B	1B			1.00	39.8%
WebMath > mix1 + mix2 > tactic	32B	1B			0.97	44.0%
Co-training (PACT)						
WebMath > mix1 + tactic	32B	18B	*0.08*		0.94	40.0%
WebMath > mix1 + mix2 + tactic	96B	71B	*0.09*	*0.09*	**0.91**	**48.4%**
Pre-training and co-training						
WebMath > mix2 > mix1 + tactic	32B	18B	*0.08*		0.93	46.9%

PACT significantly improves theorem proving pass rate on the test theorems.



Full PACT (mix1 + mix2) dominates models trained using PACT mix1 only and tactic step only across almost all modules of mathlib.

Dozens of improved proofs added to mathlib

Human-written proof

```
lemma map_bot_iff : I.map f = ⊥ ↔ I ≤
    f.ker :=
begin
  rw le_ker_iff, unfold lie_ideal.map,
    split; intros h,
  { rwa [eq_bot_iff,
    lie_submodule.lie_span_le,
    set.image_subset_iff,
    lie_submodule.bot_coe] at h,},
  { suffices : f '' I = ↑(⊥ : lie_ideal R
    L'), { rw [this,
    lie_submodule.lie_span_eq], },
    ext x, rw [lie_submodule.bot_coe,
    set.mem_singleton_iff, set.mem_image],
    split,
    { rintros ⟨y, hy, hx⟩, rw ← hx, exact h
    y hy, },
    { intros hx, use 0, simp [hx], }, },
end
```

Lean GPT-f proof

```
lemma map_bot_iff : I.map f = ⊥ ↔ I ≤
    f.ker :=
by { rw ← le_bot_iff, apply
    lie_ideal.map_le_iff_le_comap }
```

PACT improves multi-step reasoning ability

Table 1. Counting the number of semicolon-chained tactics predicted by our models that appear *in successful proofs*. Each column headed by a number n; indicates the number of times that a suggestion appeared with n occurrences of '**;**'.

Model	1;	2;	3;	4;	Mean
wm-to-tt	215	49	2	0	1.199
wm-to-tt-m1	186	39	5	1	1.225
wm-to-tt-m1-m2	**328**	**82**	**12**	**3**	**1.271**

```
theorem
  category_theory.grothendieck.congr
  {X Y : grothendieck F} {f g : X ⟶ Y}
  (h : f = g) :
  f.fiber = eq_to_hom (by subst h) ≫
  g.fiber :=
begin
  rcases X; rcases Y; subst h; simp
end
```

The Lean interactive tactic DSL uses the infix semicolon `t ; s` which will perform the tactic t and then apply the tactic s to each of the resulting subgoals produced by t. PACT improves our models' ability to predict such composite steps.