# Proving Theorem Recursively

**Haiming Wang**[1*] **Huajian Xin**[1*] Zhengying Liu[2†] Wenda Li[3]
Yinya Huang[4] Jianqiao Lu[5] Zhicheng Yang[6] Jing Tang[6,7]
Jian Yin[1†] Zhenguo Li[2] Xiaodan Liang[1,8†]

[1]Sun Yat-sen University [2]Huawei Noah's Ark Lab [3]University of Edinburgh
[4]CityU [5]HKU [6]HKUST (Guangzhou) [7]HKUST [8]Pengcheng Laboratory

# Neural theorem proving

- Theorems/proofs represented formally as **computer codes !**

**Theorem**

**Problem statement**
$\sqrt{2}$ is irrational.

**Proof**

**Proof**
Assuming $\sqrt{2} \in \mathbb{Q}$, we have $\sqrt{2}=a/b$, and $a$, $b$ are coprime. Then we have $2 = a^2/b^2$ and $2{\times}b^2 = a^2$. Thus, we know $a$ is even, $a = 2c$. Substituting $a$ into the previous equation, we have $b^2 = (2 * c)^2$. Thus, we know $b$ is also even, and $a, b$ are not coprime. This contradicts the original assumption. ■

**Verification**

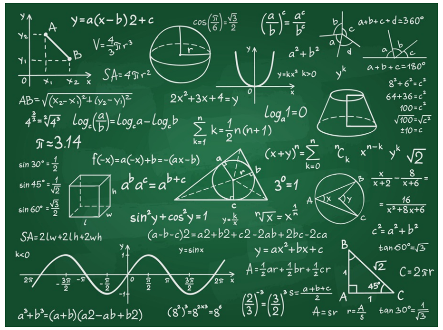
```
lemma "sqrt 2 ∉ ℚ"

proof
  assume "sqrt 2 ∈ ℚ"
  then obtain a b::int where "sqrt 2 = a/b"
    "coprime a b" "b ≠ 0" sledgehammer
  then have c: "2 = a^2 / b^2"
    sledgehammer
  then have "b^2 ≠ 0" sledgehammer
  then have *: "2*b^2 = a^2"
    sledgehammer
  then have "even a"
    sledgehammer
  then obtain c::int where "a=2*c"
    sledgehammer
  with * have "b^2 = 2*c^2"
    sledgehammer
  then have "even b"
    sledgehammer
  with ‹coprime a b› ‹even a› ‹even b›
    show False sledgehammer
qed
```

**Formal systems:**

*Coq*

*Mizar*

Sources of Theorems



Mathematics

Software

Slide from Kaiyu Yang's talk: https://mathai2023.github.io/

# Previous approaches

LM + Search (**GPT-f**, **Thor**, **DT-Solver**):

- **Language model** suggests **action** given **current state.**
- **Formal system** executes action and updates state.
- **Search algorithm** finds correct action path**.**

```
lemma "sqrt 2 ∉ ℚ"
```

goals: 1. sqrt 2 ∉ ℚ

```
proof
```

goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
assume "sqrt 2 ∈ ℚ"
```

premise: sqrt 2 ∈ ℚ
goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
then obtain a b::int where "sqrt 2 = a/b"
  "coprime a b" "b ≠ 0" sledgehammer
```

premise: sqrt 2 = real_of_int a / real_of_int b
coprime a b
b ≠ 0
goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
then have c: "2 = a^2 / b^2"
  sledgehammer
```

…

…

# Motivation

**Challenges:**

- Step-by-step methods fail to find long proofs.
- Search space grows exponentially, leading to getting lost.
- High need for value functions to guide the search.

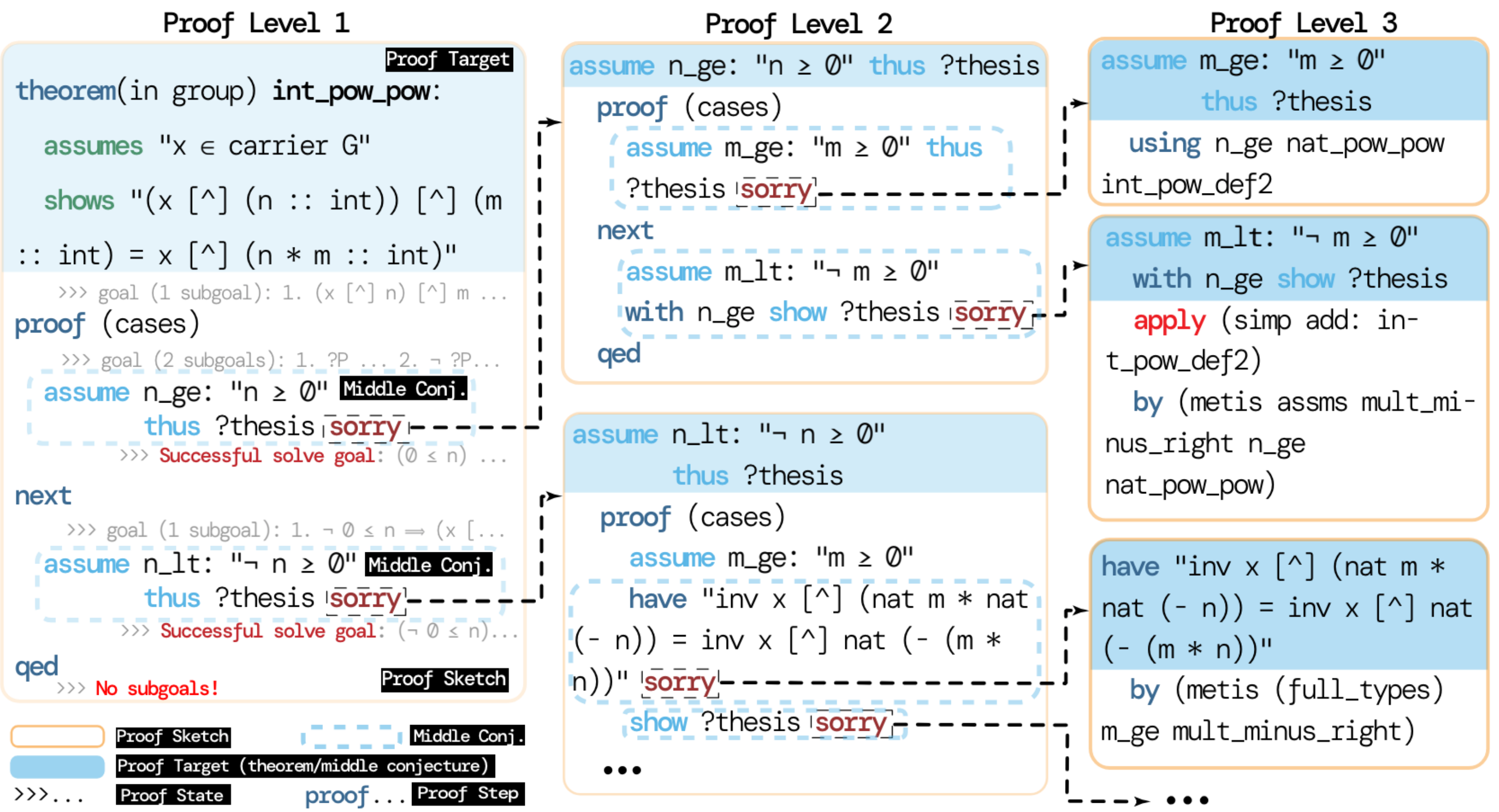
**Solution:**

- Think like humans:

  Plan → Verify → Plan → Verify → …

```
theorem(in group) int_pow_pow:
  assumes "x ∈ carrier G"
  shows "(x [^] (n :: int)) [^] (m ::
int) = x [^] (n * m :: int)"
          >>> goal (1 subgoal): 1. (x [^] n) [^] m...
proof (cases)
          >>> goal (2 subgoals): 1. ?P ⟹ (x... 2...
 assume n_ge: "n ≥ 0" thus ?thesis
          >>> using this: 0 ≤ n goal (1 subgoal):...
   proof (cases)
          >>> goal (2 subgoals): 1. 0 ≤ n ⟹  ...
     assume m_ge: "m ≥ 0" thus ?thesis
          >>> using this: 0 ≤ m goal (1 subgoal)...
      using n_ge nat_pow_pow in
t_pow_def2
          >>> Successful solve goal (m ≥ 0) ...
  ...

qed
  >>> No subgoals!                    Complete proof
```

# Prove theorem recursively




**(b) Recursive Proof**

**First recursive proving framework!**

- Search proof sketch (plan for the proof at each stage)
- Verify the proof sketch by formal system!
- Proceed to deeper sketches after verified to be correct

# Experiments: main results

### Thor (Cambridge, NeurIPS 2022)

- LM is train on single step state action pair and finds proof with best first search algorithm

### Thor + expert iteration (Google + Cambridge, NeurIPS 2022)

- Extend Thor with extensive proof data generated by Codex LLM.

### Thor + Magnushammer (Cambridge, ICLR 2023)

- Extend Thor with neural enhanced sledgehammer.

## Our approaches (proofGPT-1.3B*):

### GPT-f Baseline

- ablation setting which use step-by-step approach to prove theorem.

### POETRY

- Our recursive proving method

Table 1: **Comparing with baseline.** The table displays the pass@1 success rates of the baselines and POETRY, The highest success rates for each set are highlighted in bold.

| Success rate | miniF2F-valid | miniF2F-test | PISA | single-level | multi-level |
|---|---|---|---|---|---|
| Thor w/o sledgehammer | 25.0% | 24.2% | 39.0% | - | - |
| GPT-f Baseline | 39.3% | 37.3% | 48.9% | **65.5%** | 11.1% |
| – with sampling decoding | 30.3% | 31.5% | 43.2% | 57.8% | 9.8% |
| POETRY | **42.2%** | **42.2%** | **49.6%** | 65.4% | **13.6%** |

Table 2: **Comparing with state-of-the-art search-based methods on the miniF2F dataset.** The table displays the pass@1 success rates of previous works and POETRY, The highest success rates for each set are highlighted in bold.

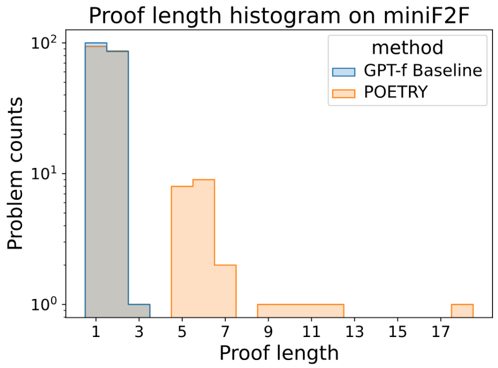
| Success rate | environment | miniF2F-valid | miniF2F-test |
|---|---|---|---|
| *Baselines* | | | |
| PACT [Han et al., 2022] | Lean | 23.9% | 24.6% |
| Leandojo [Yang et al., 2023] | Lean | - | 26.5% |
| FMSCL [Polu et al., 2022] | Lean | 33.6% | 29.6% |
| COPRA [Thakur et al., 2024] | Lean | - | 30.7% |
| Thor [Jiang et al., 2022a] | Isabelle | 28.3% | 29.9% |
| Thor + expert iteration [Wu et al., 2022] | Isabelle | 37.3% | 35.2% |
| Thor + Magnushammer [Mikuła et al., 2023] | Isabelle | 36.9% | 37.3% |
| *Ours* | | | |
| POETRY | Isabelle | **42.2%** | **42.2%** |

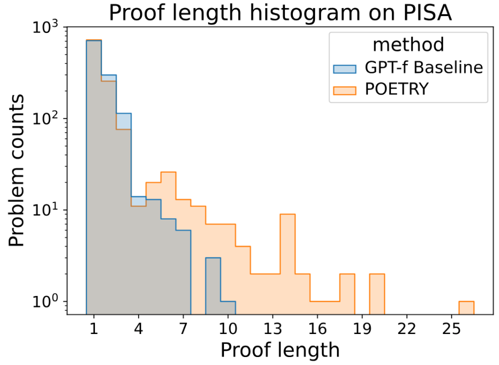
*Azerbayev, Zhangir, et al. "Proofnet: Autoformalizing and formally proving undergraduate-level mathematics." *arXiv preprint arXiv:2302.12433* (2023).

# Experiments: analysis

## POETRY capable of finding long proof

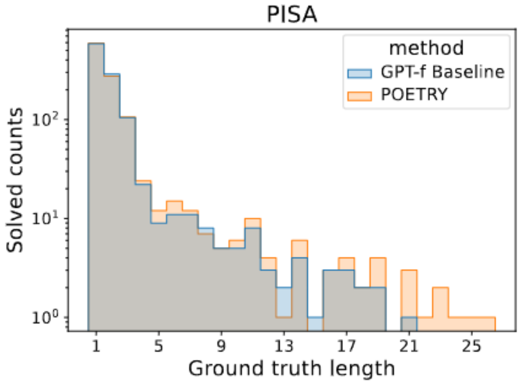
- GPT-f Baseline found maximum length of 3 (10) steps in miniF2F (PISA)
- POETRY found maximum length of 18 (26) steps in miniF2F (PISA)

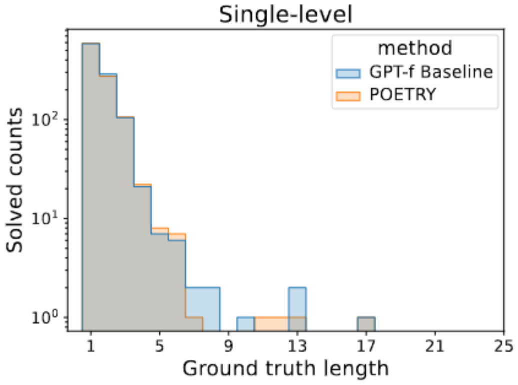


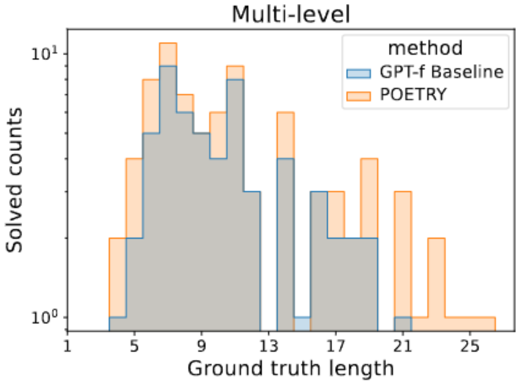




## POETRY capable of finding harder proof

- POETRY and GPT-f Baseline have similar performance in single-level problem
- POETRY excels at solving problems requires structural reasoning.



# Experiments: case study

```
lemma(in UP_cring) n_mult_closed:
  assumes "ƒ ∈ carrier P"
  shows "n_mult ƒ ∈ carrier P"
proof(rule UP_car_memI[of "deg R ƒ"])
  show "⋀n. deg R ƒ < n ⟹ n_mult ƒ n = 0"
    unfolding n_mult_def
    using assms
    unfolding P_def
    by (simp add: UP_car_memE(2))          Proof level 2
  show "⋀n. n_mult ƒ n ∈ carrier R"
    using assms
    unfolding n_mult_def
    by (simp add: assms cfs_closed)        Proof level 2
qed                                        Proof level 1
```

(a)

```
        lemma(in UP_cring) n_mult_closed:
          assumes "ƒ ∈ carrier P"
          shows "n_mult ƒ ∈ carrier P"
Path 1  proof(rule UP_car_memI[of "deg R ƒ"])
          fix n
          assume A: "deg R ƒ < n"
          show "n_mult ƒ n = 0"
            unfolding n_mult_def
            proof -
           ✗ Timeout after 600 seconds

Path 2  proof(rule UP_car_memI[of "deg R ƒ"])
          show "⋀n. deg R ƒ < n ⟹ n_mult ƒ n = 0"
           ✗ Never explored
```

(b)

**Case comparison between POETRY and GPT-f Baseline.**

- Recursive proof found by POETRY in 71.2 seconds, the proof contains two proof levels.
- Failure-proof paths found by the GPT-f Baseline. GPT-f Baseline failed to find proof due to timeout after 600 seconds. We select two different failure proof paths found by GPT-f Baseline.

# Thanks