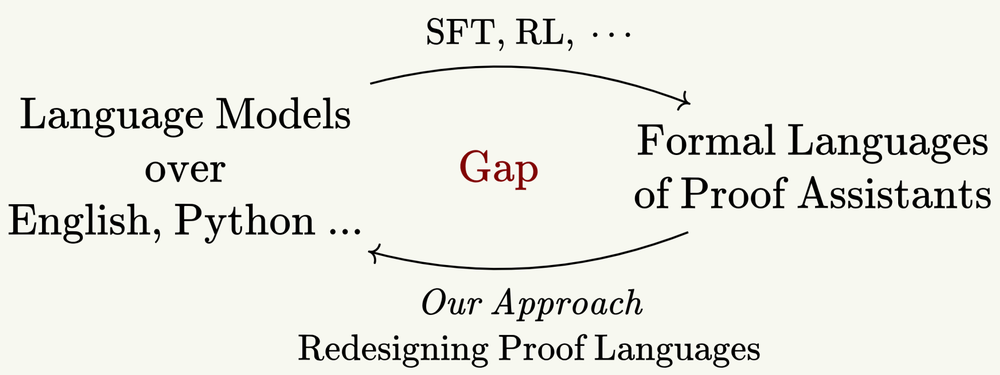
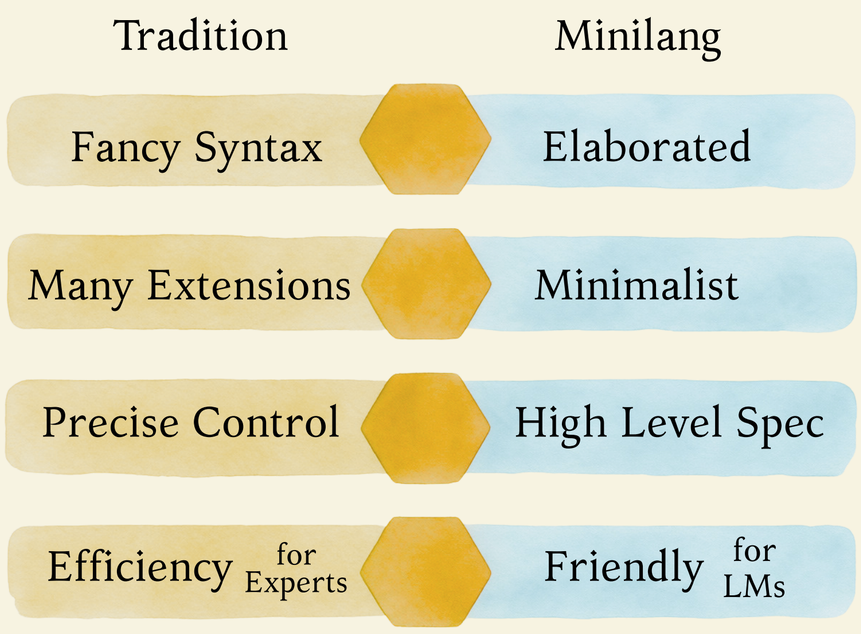
A Minimalist Proof Language for Neural Theorem Proving over Isabelle/HOL

Qiyuan Xu[1], Renxi Wang[2], Peixin Wang[3], Haonan Li[2], Conrad Watt[1]
Nanyang Technological University[1], MBZUAI[2], China East Normal University[3]

01 Introduction

A bottleneck in Neural Theorem Proving is the gap between NL and FL. Training/Finetuning (SFT, RL) pushes LMs closer to PAs while our approach redesign FLs to approach LMs





02 Minilang

Minilang (below) preserves declarative structures and removes tactics and expert features from Isar (right)

```
theorem sqrt2_not_rational: "√2 ∉ ℚ"
  RULE proof_by_contradiction
  CONSIDER "∃mn. |√2| = m/n ∧ coprime m n" END
  HAVE "m^2 = (√2)^2 * n^2" END
  HAVE eq: "m^2 = 2 * n^2" END
  HAVE "2 dvd m^2" END WITH eq
  HAVE "2 dvd m" END
  HAVE "2 dvd n"
    CONSIDER k where "m = 2 * k" END
    HAVE "2 * n^2 = 2^2 * k^2" END WITH eq
    HAVE "2 dvd n^2" END
  END
  HAVE "2 dvd gcd m n" END
  HAVE "2 dvd 1" END
END
```

```
theorem sqrt2_not_rational: "sqrt 2 ∉ ℚ"
proof
  let ?x = "sqrt 2"
  assume "?x ∈ ℚ"
  then obtain m n :: nat where
    "|?x| = m / n" and "coprime m n"
    by (rule Rats_abs_nat_div_natE)
  hence "m^2 = ?x^2 * n^2"
    by (auto simp add: power2_eq_square)
  hence eq: "m^2 = 2 * n^2"
    using of_nat_eq_iff power2_eq_square by force
  hence "2 dvd m^2" by simp
  hence "2 dvd m" by simp
  have "2 dvd n" proof -
    from "2 dvd m" obtain k where "m = 2 * k" ..
    with eq have "2 * n^2 = 2^2 * k^2" by simp
    hence "2 dvd n^2" by simp
    thus "2 dvd n" by simp
  qed
  with "2 dvd m" have "2 dvd gcd m n"
    by (rule gcd_greatest)
  with lowest_terms have "2 dvd 1" by simp
  thus False using odd_one by blast
qed
```

Minilang restricts its (core) language constructs:

HAVE decomposing a proof goal into step-by-step subgoals.

CONSIDER$_\vee$ analyzing a proof goal by cases, e.g., consider the cases where x is positive, zero, or negative.

CONSIDER$_\exists$ binding variables to the witnesses of existential statements, e.g., consider a number p such that p is a prime greater than 2025.

Proof operations commonly found in informal proofs

RULE proving a goal by a specific mode of argument, e.g., arguing by contradiction for a given goal, and deriving $A \longrightarrow B$ and $B \longrightarrow A$ to show $A \longleftrightarrow B$.

CHOOSE proving an existential statement by providing a witness.

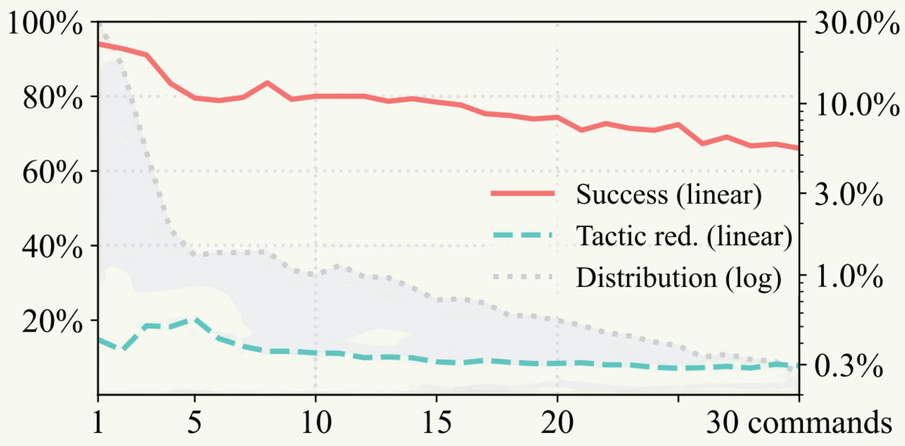
SIMPLIFY equivalently rewriting the proof goal into a simpler form.

CASE_SPLIT applying structural case analysis to the goal.

INDUCT applying induction to the goal.

END WITH *ps* indicates that the target proof goal straightforwardly follows from the given premises *ps*.

03 Translation & Experiments



We build a translator to map ~5M lines of Isar into Minilang serving as our training corpus. We then evaluate SFT on PISA

Base Model	Language	ATP	pass@1	pass@4	pass@8
DPSK-PB	Minilang	SH*	**69.1%**	**76.0%**	**79.2%**
DPSK-PB	Minilang	SH	59.9%	68.1%	72.4%
DPSK-PB	Thor-style Isar	SH*	63.9%	69.6%	74.3%
DPSK-PB	Thor-style Isar	SH	45.7%	54.9%	59.7%
DPSK-PB	Minilang	none	35.5%	40.6%	44.9%
DPSK-PB	Isar	none	40.2%	45.8%	50.5%
Llemma	Minilang	SH*	**68.0%**	**74.9%**	**78.9%**
Llemma	Minilang	SH	58.7%	67.9%	72.2%
Llemma	Thor-style Isar	SH*	63.3%	66.9%	72.1%
Llemma	Thor-style Isar	SH	46.1%	51.9%	57.5%
Llemma	Minilang	none	35.2%	39.8%	44.6%
Llemma	Isar	none	38.6%	43.9%	48.6%

04 Agent over AST

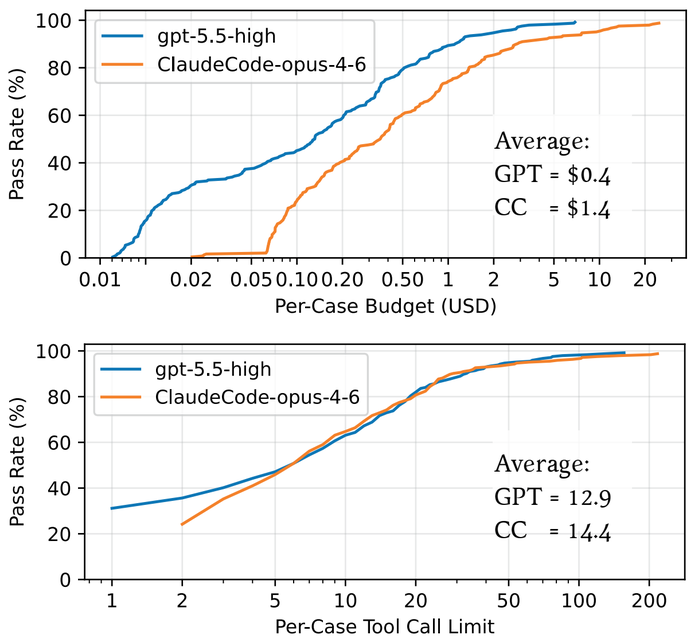
Conventional Proof Agents:
1. LMs generate source in a concrete syntax.
2. LMs emit thousands of LSP calls to retrieve information separately

Minilang's AoA Agent
1. Deprecate proof languages' concrete syntax and represent Minilang's AST in whatever format LMs feel comfortable, e.g., JSON and YAML.
2. Redesign the interface to integrate all the information required.

Tool Calling Argument

```
{
  "action": "fill",
  "target_step": "1",
  "proof_operations": [
    { "operation": "Contradiction",
      "hypothesis_name": "h"},
    { "operation": "Obtain",
      "variables": [
        { "name": "m", "type": "nat" },
        { "name": "n", "type": "nat" } ],
      "constraints": [
        { "name": "sqrt_frac",
          "statement": {
            "english": "the square root equals the fraction m/n",
            "isabelle": "¦sqrt (2::real)¦ = real m / real n" }
        }, {
          "name": "cop",
          "statement": {
            "english": "m and n are coprime",
            "isabelle": "coprime m n" } } ],
    "proof": [
      { "operation": "Obvious" },
        ...
```

Evaluation on MiniF2F



Tool Calling Return

```
Outline:
  step 1: Contradiction (hypothesis: h)
  step 2: Obtain m, n (done)
    obtained variables:
      - m: nat
      - n: nat
    constraints:
      - n_nonzero: n ≠ (0 :: nat)
      - sqrt_frac: ¦sqrt (2 :: real)¦ = real m / real n
      - cop: coprime m n
  Error: Unfinished Proof. Fill step `3`
```