LEGO-Prover: Neural Theorem Proving with Growing Libraries

Haiming Wang[1*] Huajian Xin[1*] Chuanyang Zheng[3] Zhengying Liu[2†]
Qingxing Cao[1] Yinya Huang[4] Jing Xiong[1] Han Shi[2] Enze Xie[2]
Jian Yin[1†] Zhenguo Li[2] Xiaodan Liang[1,5,6†]

[1]Sun Yat-sen University [2]Huawei Noah's Ark Lab [3]The Chinese University of Hong Kong
[4]City University of Hong Kong [5]MBZUAI [6]DarkMatter AI Research

Automated Theorem Proving

Problem statement

$\sqrt{2}$ is irrational.

```
lemma "sqrt 2 ∉ ℚ"
```

Prover

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. ■

```
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 system Isabelle

LM + Search (**GPT-f** OpenAI 2021, **Thor** Cambridge 2021, **DT-Solver** Ours 2023):

- **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
```

...

...

LLM with ICL (**DSP** Cambridge 2022, **Subgoal-based** HKU 2023):

- **ChatGPT (gpt-3.5-turbo)** generates **entire proof in one go**, with in-context learning
- **Formal System** verifies the proof

```
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
```

No goals !

Error: xxx

Automated theorem proving:

- Verifiable
- Longer reasoning chain
- Data scarcity

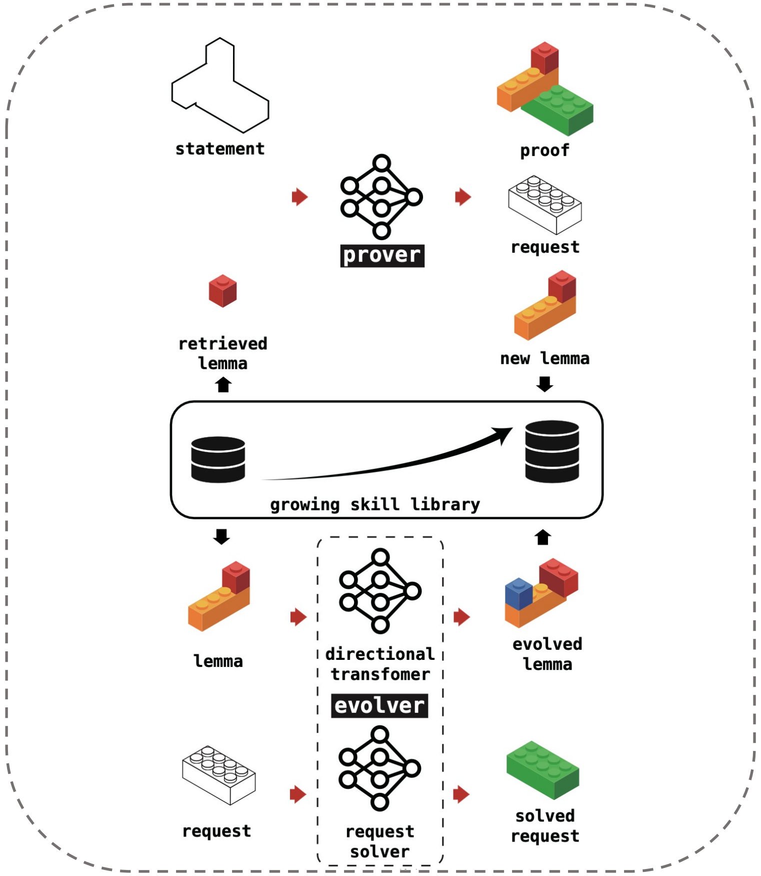
Motivation

- Problems with existing provers:
  - Each theorem is proved **independently**.
  - Proven conjectures are **not shared** among problems.
  - LLM struggles to generate **correct long-chain proof** (hallucination).

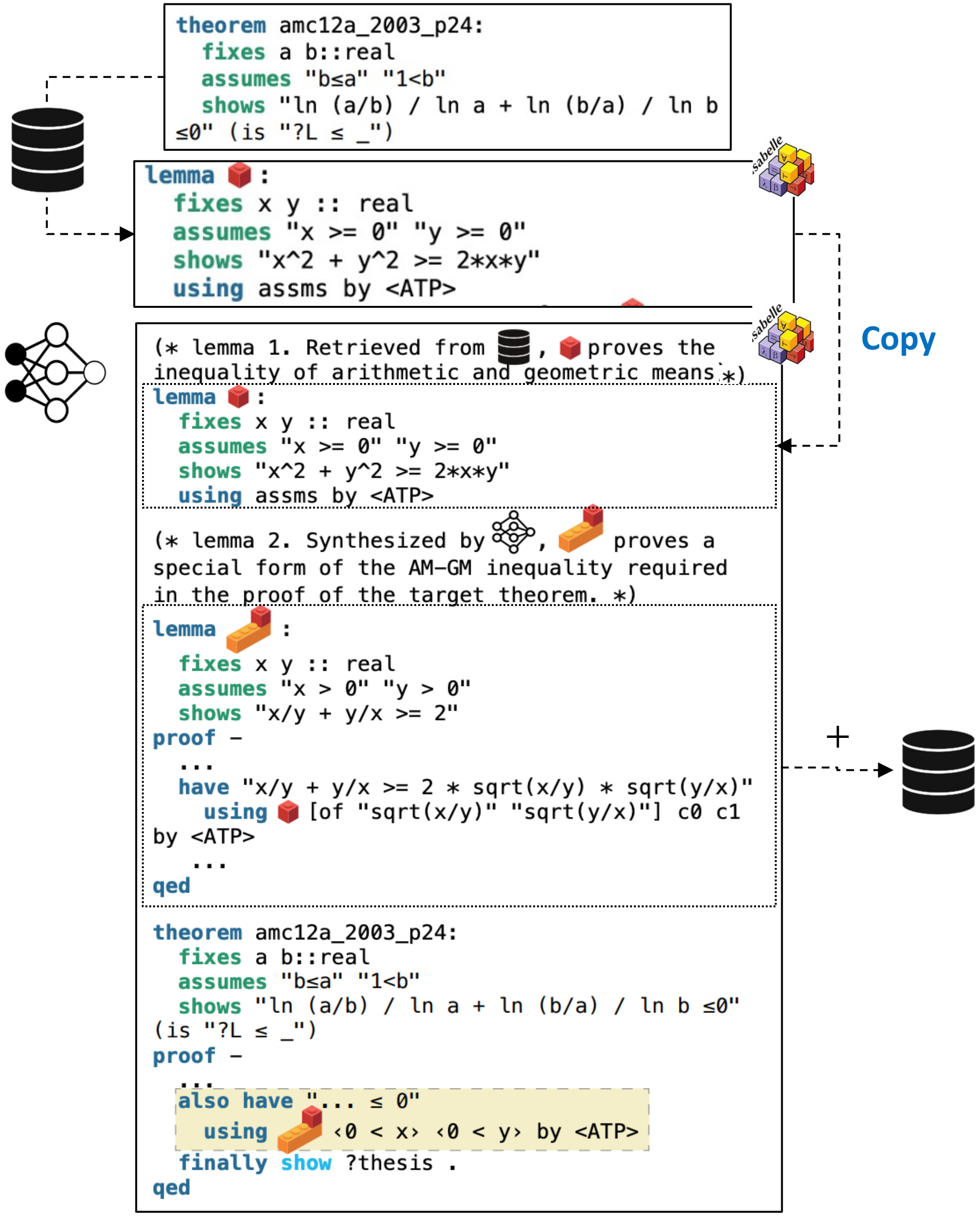
- Ideal provers:
  - Extract & **reuse** useful lemmas during each theorem proving, **to reduce reasoning length**
  - Maintain & **grow** a library of proven theorems/lemmas (online & offline)
  - Leverage the power of LLM (prover)
  - Leverage the verification capability of formal systems (Lean, Isabelle)
  - Imitate human proving process

LEGO-Prover: Prove Theorem Like Building LEGO

Prove in a **block-by-block manner**
- Prove **sub-goal lemmas**
- Prove theorem using sub-goal lemmas.
- Sub-goal Lemmas: retrieved from skill library, or constructed online



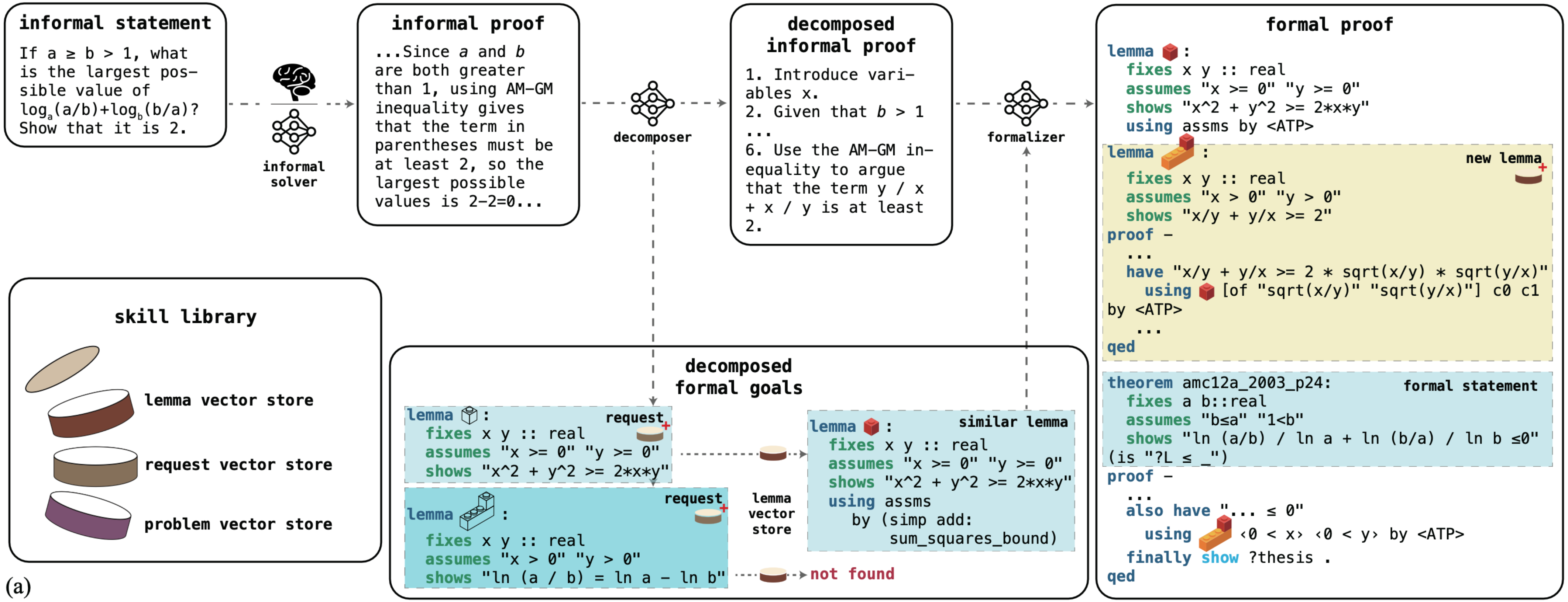
LEGO-Prover consists of a prover, an evolver, and a growing skill library



LEGO-Prover: Prover

Three proof steps

- **Informal solver:** produce an informal proof
- **Decomposer:** produce step-by-step informal proof and sub-goals lemma statements, which are used to retrieve useful lemma from the skill library.
- **Formalizer:** prove theorem with step-by-step informal proof and retrieved lemmas block-by-block.

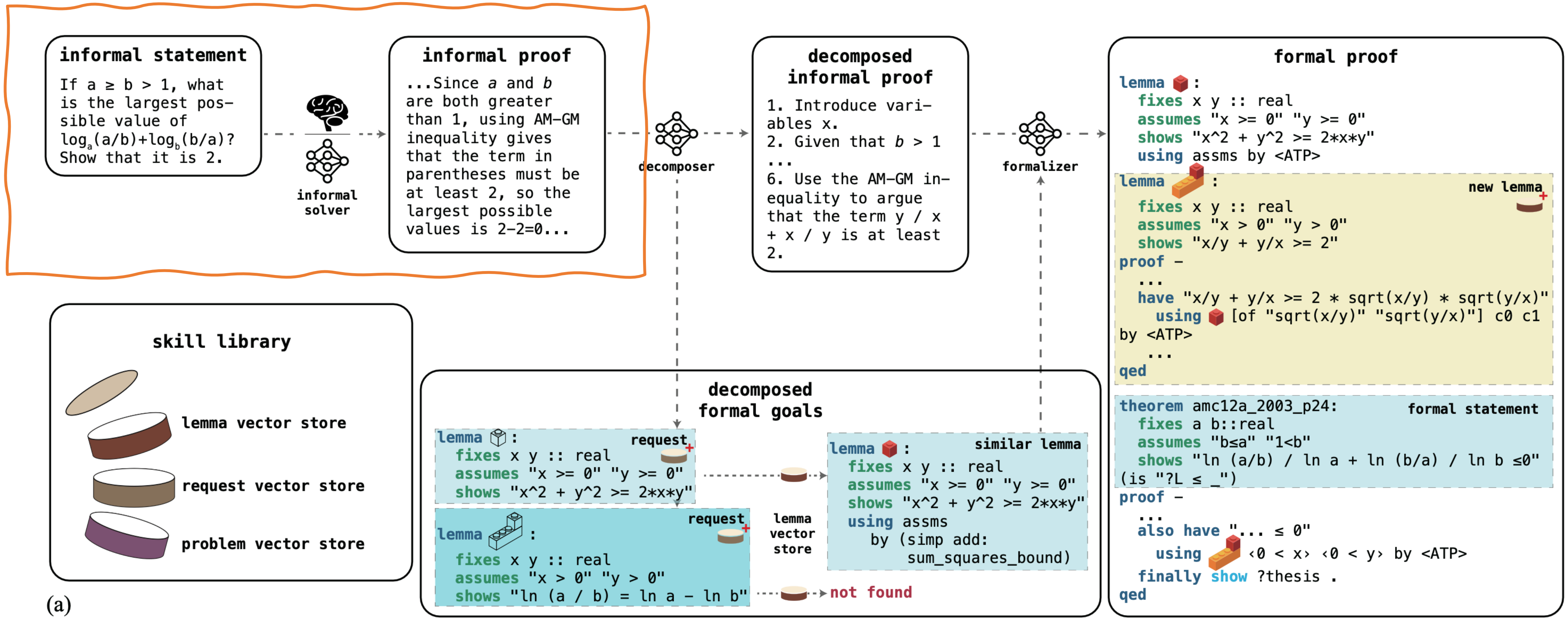


(a)

LEGO-Prover: Prover

Three proof steps

- **Informal solver:** produce an informal proof
- **Decomposer:** produce step-by-step informal proof and sub-goals lemma statements, which are used to retrieve useful lemma from the skill library.
- **Formalizer:** prove theorem with step-by-step informal proof and retrieved lemmas block-by-block.

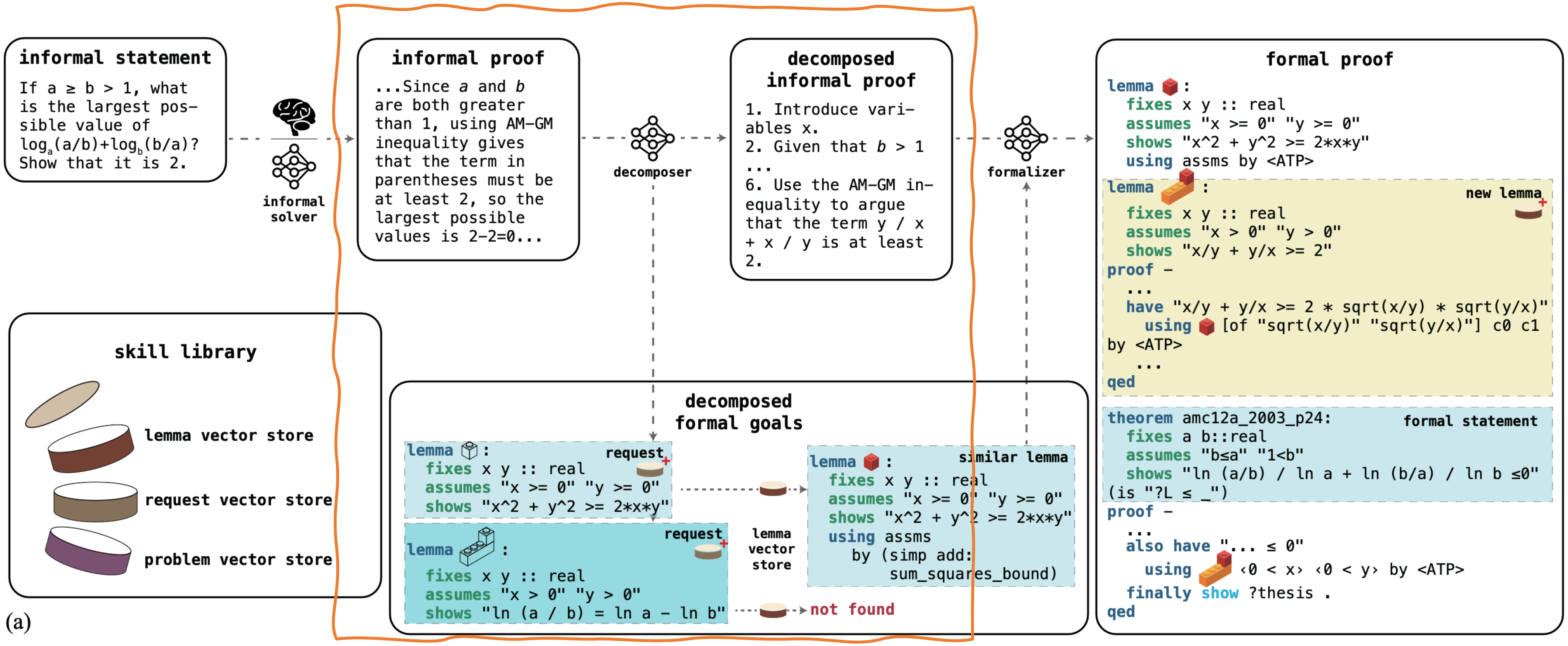


(a)

LEGO-Prover: Prover

Three proof steps

- **Informal solver:** produce an informal proof
- **Decomposer:** produce step-by-step informal proof and sub-goals lemma statements, which are used to retrieve useful lemma from the skill library.
- **Formalizer:** prove theorem with step-by-step informal proof and retrieved lemmas block-by-block.

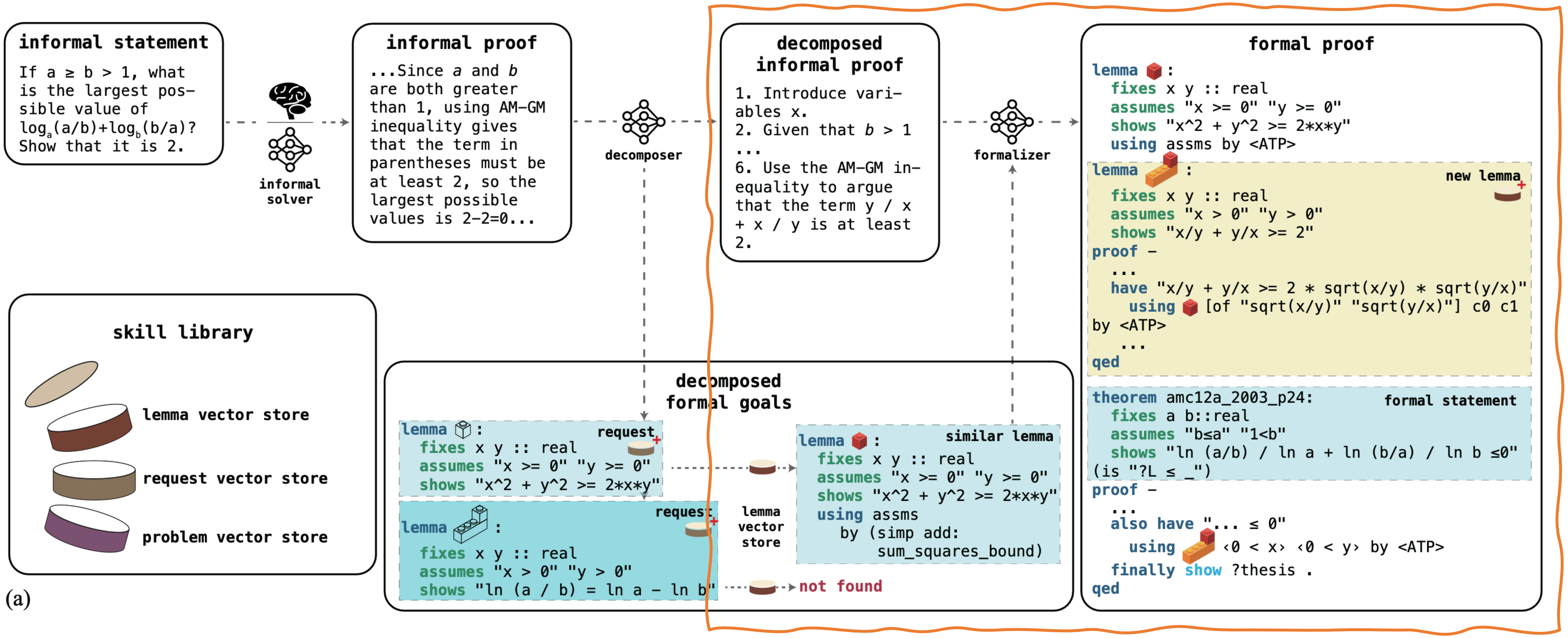


(a)

LEGO-Prover: Prover

Three proof steps

- **Informal solver:** produce an informal proof
- **Decomposer:** produce step-by-step informal proof and sub-goals lemma statements, which are used to retrieve useful lemma from the skill library.
- **Formalizer:** prove theorem with step-by-step informal proof and retrieved lemmas block-by-block.



(a)

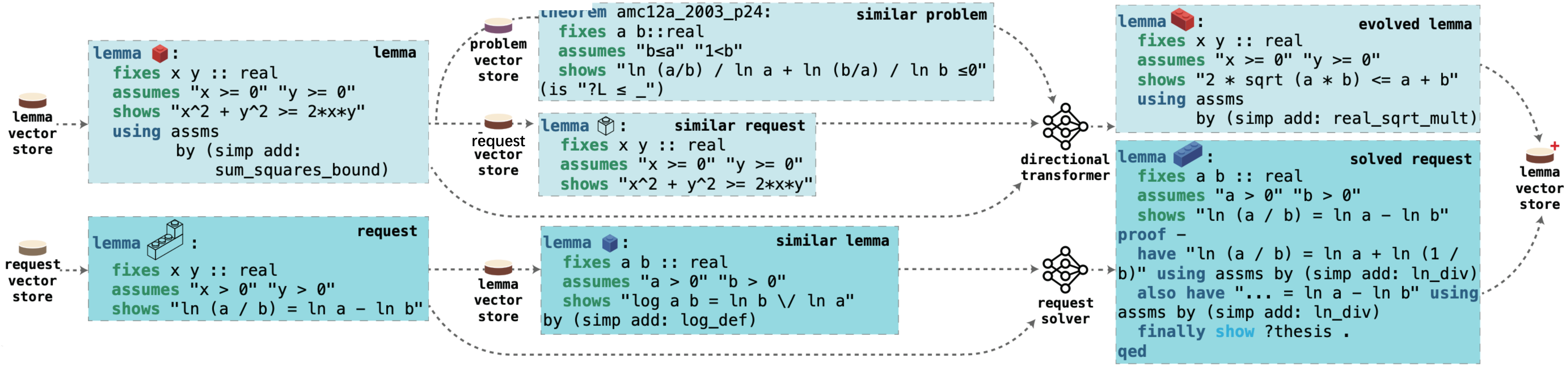
LEGO-Prover: Evolver

Transforms existing skills into a more general and reusable form, or directly solves requested subgoals proposed by the prover.

- **Directional transformer** evolves skill using four type of specific direction
- **Request solver** directly solves the request proposed by the decomposer.

Different types of directional transformer

Evolve type	Description
Identify key concepts	Determine the essential ideas, methods, or theorems that are crucial to solving the initial problem.
Parameterize	If the problem involves specific numbers, generalize it by replacing these with variables.
Scale complexity	Try both simpler and more complicated versions of the problem to see how the approach adapts.
Extend dimensions	If the problem is defined in a specific number of dimensions, consider if it holds in more or fewer dimensions.



Experiments

- **Thor (Cambridge, NeurIPS 2022)**: LM + Search. LM trained on single step state-action pairs. Find proof with best first search.
- **Thor + expert iteration (Google + Cambridge, NeurIPS 2022)**: LM + Search. Extend Thor with extensive data by Codex.
- **DSP (Cambridge, ICLR 2023)**: LLM with ICL, use informal proof to guide Codex to generate formal sketch.
- **Subgoal-Learning (HKU + Cambridge, NeurIPS 2023)**: LLM with ICL, extends DSP with step-by-step informal proof.

Success rate	LLM	miniF2F-valid	miniF2F-test
Baselines			
Thor (Jiang et al., 2022a)	-	28.3%	29.9%
Thor + expert iteration (Wu et al., 2022)	Codex	37.3%	35.2%
Draft, sketch, and Prove (Jiang et al., 2022b)	Codex	42.6%	39.3%
Subgoal-Learning (Zhao et al., 2023)	ChatGPT	48.0%	45.5%
Ours (100 attempts)			
LEGO-Prover (model informal proof)	ChatGPT	52.4%	45.5%
LEGO-Prover (human informal proof)	ChatGPT	55.3%	**50.0%**
LEGO-Prover*	ChatGPT	**57.0%**	**50.0%**
Ablations (100 attempts)			
- Skill Library (human informal proof)	ChatGPT	50.4%(−4.9%)	-

Experiments: Case Study

retrieved skill

```
lemma am_gm:
  fixes x :: real
  assumes "x > 0"
  shows "x + 1/(2*x) ≥ sqrt 2"
proof -
  have "(sqrt x - sqrt (1/(2*x)))^2 ≥ 0" by simp
  then have "x + 1/(2*x) - 2 * sqrt x * sqrt (1/(2*x)) ≥ 0"
    by (smt (verit) add_le_imp_le_diff assms divide_nonneg_nonneg less_eq_-
real_def real_sqrt_pow2 sum_squares_bound)
  then have "x + 1/(2*x) - sqrt 2 ≥ 0"
    by (metis add_divide_distrib assms div_by_1 divide_divide_eq_right di-
vide_self less_numeral_extra(3) mult_2 one_add_one real_div_sqrt re-
al_sqrt_divide real_sqrt_eq_1_iff real_sqrt_gt_0_iff real_sqrt_one times_di-
vide_eq_right zero_le_numeral)
  then have "x + 1/(2*x) ≥ sqrt 2" by simp
  then show ?thesis by simp
qed
```

synthesized proof

```
lemma am_gm:
  fixes x :: real
  assumes "x > 0"
  shows "x + 1/(2*x) ≥ sqrt 2"
proof -
  have "(sqrt x - sqrt (1/(2*x)))^2 ≥ 0" by simp
  then have "x + 1/(2*x) - 2 * sqrt x * sqrt (1/(2*x)) ≥ 0"
    by (smt (verit) add_le_imp_le_diff assms divide_nonneg_nonneg less_eq_-
real_def real_sqrt_pow2 sum_squares_bound)
  then have "x + 1/(2*x) - sqrt 2 ≥ 0"
    by (metis add_divide_distrib assms div_by_1 divide_divide_eq_right di-
vide_self less_numeral_extra(3) mult_2 one_add_one real_div_sqrt re-
al_sqrt_divide real_sqrt_eq_1_iff real_sqrt_gt_0_iff real_sqrt_one times_di-
vide_eq_right zero_le_numeral)
  then have "x + 1/(2*x) ≥ sqrt 2" by simp
  then show ?thesis by simp
qed

theorem algebra_amgm_faxinrrp2msqrt2geq2mxm1div2x:
  "⋀x. (x>0) ⟹ 2 - sqrt 2 ≥ 2 - x - 1/ (2 * x)"
proof -
  fixes x :: real
  assumes h0: "x>0"
  (*Step 1: We want to show that $x + \frac{1}{2x} \geq \sqrt{2}$ is true.*)
  (* Step 2: Apply the AM-GM inequality to $x$ and $\frac{1}{2x}$ to get $x
+ \frac{1}{2x} \geq 2\sqrt{\frac{x}{2x}}$. *)
  (* Step 3: Simplify $2\sqrt{\frac{x}{2x}}$ to $\sqrt{2}$. *)
  (* Step 4: Conclude that $x + \frac{1}{2x} \geq \sqrt{2}$ is true. *)
have c1: "f (500 * (6/5)) = f 500 / (6/5)"
  have "x + 1/(2*x) ≥ sqrt 2" using am_gm[OF h0] by simp
  (* Step 5: Since $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$ is equivalent to
$x + \frac{1}{2x} \geq \sqrt{2}$, we can conclude that $2 - \sqrt{2} \geq 2
- x - \frac{1}{2x}$ is true. *)
  then show "2 - sqrt 2 ≥ 2 - x - 1/ (2 * x)" by simp
qed
```

(a) Directly Use

Retrieved skill:

lemma am_gm: For a real number x, $x > 0$, prove that $x + \frac{1}{2x} \geq \sqrt{2}$.

Proof. We have $\left(\sqrt{x} + \sqrt{\frac{1}{2x}}\right)^2 \geq 0$. Expanding the inequality, we obtain $x + \frac{1}{2x} - 2 * \sqrt{x} * \sqrt{\frac{1}{2x}} \geq 0$. From which we have $x + \frac{1}{2x} - \sqrt{2} \geq 0$, and thus $x + \frac{1}{2x} \geq \sqrt{2}$. ■

copy paste by LLM

Synthesized proof:

lemma am_gm: For a real number x, $x > 0$, prove that $x + \frac{1}{2x} \geq \sqrt{2}$.

Proof. We have $\left(\sqrt{x} + \sqrt{\frac{1}{2x}}\right)^2 \geq 0$. Expanding the inequality, we obtain $x + \frac{1}{2x} - 2 * \sqrt{x} * \sqrt{\frac{1}{2x}} \geq 0$. From which we have $x + \frac{1}{2x} - \sqrt{2} \geq 0$, and thus $x + \frac{1}{2x} \geq \sqrt{2}$. ■

theorem algebra_amgm_faxinrrp: Given a real number x, prove that the expression $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$ holds true for all $x > 0$.

Proof. Using the proven lemma **am_gm,** we can show that $x + \frac{1}{2x} \geq \sqrt{2}$. Multiplying both sides with -1 and add 2, we obtain $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$. ■

Case **directly use**:

- A verified lemma am_gm is retrieved from skill libraries (with proof).
- Formalizer synthesized final proof using retrieved skill directly.

  1) Copy pasted the lemma **am_gm** in the proof code directly.

  2) Prove main theorem using the proven **am_gm** lemma.

Experiments: Case Study

Case **propose lemma by imitation**:

- A verified lemma **prod_1n_4n** is retrieved from skill libraries (proof).
- Formalizer synthesized final proof by solving the lemma imitating the retrieved skill.

  1) Imitate the lemma **prod_1n_4n**. The formalizer uses induction to prove **prod_frac_common_factor.**

  2) Prove main theorem using the proven **prod_frac_common_factor** lemma**.**

Retrieved skill:

lemma prod_1n_4n: for a natural number n, prove that $\prod_1^n 4 * k = 4^n * n!$

Proof. Let's prove by induction on n. For the base case we have $\prod_1^1 4 * k = 4^1 * 1!$, we have $4 = 4$. For induction step, assuming $\prod_1^j 4 * k = 4^j * j!$, we prove that $\prod_1^{j+1} 4 * k = 4^{j+1} * (j+1)!$. Since $\prod_1^{j+1} 4 * k = 4^j * j! * (4 * (j+1))$, thus $\prod_1^{j+1} 4 * k = 4^{j+1} * (j+1)!$. ■

Propose lemma by imitation

Synthesized proof:

lemma prod_frac_common_factor: for a natural number n and a real number a, prove that $\prod_1^n \frac{ak+a}{ak} = n+1$

Proof. Let's prove by induction on n. For the base case we have $\prod_1^1 \frac{ak+a}{ak} = \frac{a*1+a}{a*1} = 1+1$. For induction step, we have $\prod_1^{n+1} \frac{ak+a}{ak} = \frac{a*(n+1)+a}{a*(n+1)} * \prod_1^n \frac{ak+a}{ak} = \frac{a*(n+1)+a}{a*(n+1)} * n+1 = n+1+1$. ■

theorem amc12a_2008_p4: Prove that $\prod_1^{501} \frac{4k+4}{4k} = 502$.

Proof. Using the proven lemma **prod_frac_common_factor,** we can show that $\prod_1^{501} \frac{4k+4}{4k} = 501 + 1 = 502$. ■

retrieved skill

```
lemma prod_1n_4n:
  fixes n :: nat
  shows "(∏k = 1..n. (4 ∗ k)) = 4^n ∗ fact n"
proof(induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  then show ?case by (simp add: algebra_simps)
qed
```

synthesized proof

```
lemma prod_frac_common_factor:
  fixes n:: nat and a::real
  assumes "a ≠ 0"
  shows "(∏k = 1..n. (a ∗ k + a) / (a ∗ k)) = n + 1"
proof(induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  have "(∏k = 1..Suc n. (a ∗ k + a) / (a ∗ k)) = ((a ∗ Suc n + a) / (a ∗ Suc
n)) ∗ (∏k = 1..n. (a ∗ k + a) / (a ∗ k))"
    by simp
  also have "... = ((a ∗ Suc n + a) / (a ∗ Suc n)) ∗ (n + 1)"
    using Suc by simp
  also have "... = Suc n + 1"
    using assms by (smt (verit) add.commute add_divide_distrib divisors_zero
nonzero_mult_div_cancel_left nonzero_mult_divide_mult_cancel_right of_nat_1
of_nat_add of_nat_neq_0 plus_1_eq_Suc right_inverse_eq times_divide_eq_left)
  finally show ?case by simp
qed

theorem amc12a_2008_p4:
  "(∏k::nat=1..501. ((4::real) ∗ k + 4) / (4 ∗ k)) = 502"
proof -
  (∗ Step 1: Rewrite the given product as $\frac {8}{4}\cdot\frac {12}{8}\c-
dot\frac {16}{12}\cdots\frac {4n + 4}{4n}\cdots\frac {2008}{2004}$. ∗)
  have "(∏k::nat=1..501. ((4::real) ∗ k + 4) / (4 ∗ k)) = (∏k::nat=1..501.
(4 ∗ (k + 1)) / (4 ∗ k))"
    by eval
  (∗ Step 2: Simplify the product by canceling out common factors. Notice
that each term in the numerator cancels with the corresponding term in the
denominator, leaving only the last term $\frac {2008}{4}$. ∗)
  also have "... = (∏k::nat=1..501. (k + 1) / k)"
    by eval
  (∗ Use lemma 1 to simplify the product ∗)
  also have "... = 501 + 1"
    using prod_frac_common_factor[of "1::real" "501"] by eval
  (∗ Step 3: Calculate the value of $\frac {2008}{4}$ to find that it is
equal to $502$. ∗)
  also have "... = 502"
    by simp
  (∗ Step 4: Conclude that the given product is equal to $502$. ∗)
  finally show ?thesis by simp
qed
```

(b) Propose Lemma by Imitation

Conclusion

1. We proposed LEGO-Prover, a novel method for automated theorem proving, which utilizes a growing skill library to construct proof in a modularity way.

2. The learned skill library serves as a valuable enhancement on the standard Isabelle library, which includes many useful high-level lemmas that are useful for other problems.

3. LEGO-Prover advances the state-of-the-art pass rate on miniF2F-valid (48.0% to 57.0%) and miniF2F-test (45.5% to 50.0%)