

The Curious Landscape of LLM Reasoning

An Overview of Techniques, Research Directions, and Challenges

Bohan Lyu

`bohanlyu.com`

Tsinghua University

October 25, 2025

Dual Degree: Computer Science & Finance, 2022-2026.

2023.7 China Securities.

2023.9 THUNLP.

2024 Summer UCSD.

2024.10-Present Princeton.

Pls refer to <https://bohanlyu.com> for more details.

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Chain of Thought

CoT: Chain of Thought, “Let’s think step by step”.

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Figure: Prompting Chain of Thought.

OpenAI-O1, DeepSeek-R1.

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\dots$

$$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

$\dots$

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\dots$

We started with the equation:

$$\sqrt{a - \sqrt{a + x}} = x$$

First, let's square both sides:

$$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$$

Next, I could square both sides again, treating the equation: $\dots$

$\dots$

Figure: The “Aha Moment” observed by DeepSeek-R1.

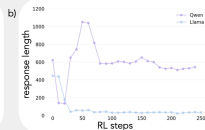
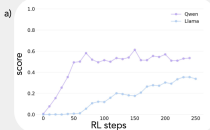
Why can DeepSeek-R1 succeed? (1/2)

- Reinforcement learning based on the base model (with warm-up).
- Reflection behaviors in the base model.

A tale of two models: **Qwen 2.5 3B** and **Llama 3.2 3B**

Let's start with the sum of the largest two numbers and then subtract the smallest two: $84 + 83 - 34 - 72$. This gives us $167 - 76$ which equals 91 . That's not 39 . Let's try another combination: $84 + 83 - 72 - 34$. This gives us $167 - 106$ which equals 61 . That's still not 39 . Let's try $84 + 72 - 83 - 34$. This gives us $156 - 117$ which equals 39 . This is the correct equation.

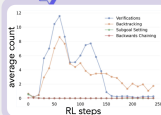
84 is the difference between 108 and 34.
<answer> $(84 - 34) / 108$ </answer>



A contrast in behaviors explored by the two models

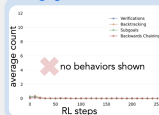
Verifications
"Let me check my answer ..."

Subgoal Setting
"Let's try to get to a multiple of 10"

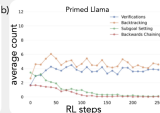
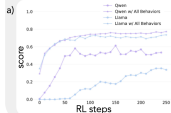


Backtracking
"Let's try a different approach, what if we ..."

Backward Chaining
"Working backwards, 24 is 8 times 3"



Priming with behaviors reduces performance gap

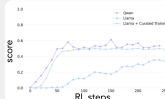


We can curate a continued pre-training set so that Llama shows similar improvements to Qwen

OpenWebMath

Filter for behaviors

Reformat as:
Query, Thought,
Answer



Why can DeepSeek-R1 succeed? (2/2)

What else?

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCodeBench
	pass@1	cons@64	pass@1	pass@1	pass@1
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9
DeepSeek-R1-Zero-Qwen-32B	47.0	60.0	91.6	55.0	40.2
DeepSeek-R1-Distill-Qwen-32B	72.6	83.3	94.3	62.1	57.2

Table: Comparison of distilled and RL Models on Reasoning-Related Benchmarks.

Rethink the position of academia in the development of LLM.

Rethink our position in academia.

“The Role of Computing Resources in Publishing Foundation Model Research”

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

An Overview

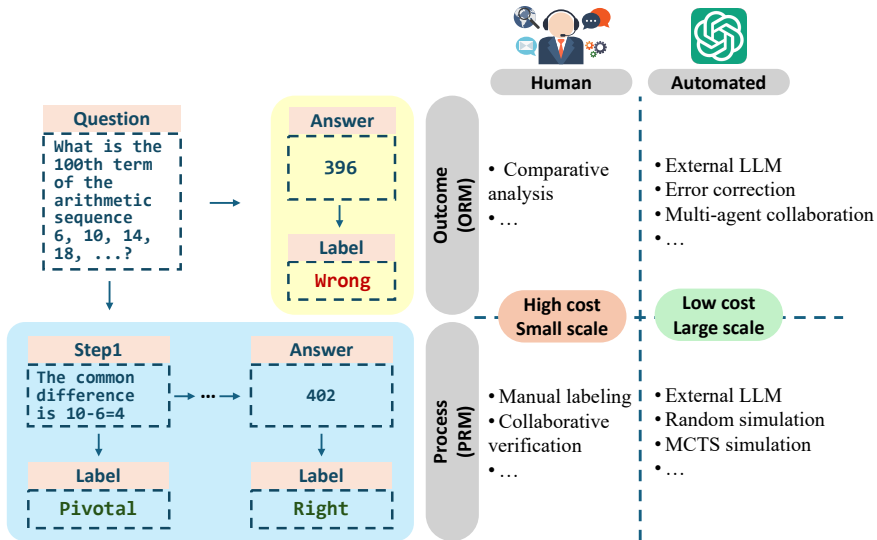
What are the techniques that can boost LLM Reasoning?

- Supervised Fine-Tuning (SFT) based methods: data is the key.
 - Human Annotation, LLM Distillation.
 - Self-Evolve: Reject Sampling Fine-tuning.
- Reinforcement Learning (RL) based methods:
 - Data: Filtering by Difficulty, Curriculum Learning.
 - Pipeline: Reinforcement Learning with Human Feedback (RLHF)&Reinforcement Learning with Verifiable Reward (RLVR), Process Reward Model (PRM)&Outcome Reward Model (ORM).
 - RL Algorithm: Proximal Policy Optimization (PPO), Group Relative Policy Optimization (GRPO), REINFORCE++.
- Direct Preference Optimization (DPO) and its variants, e.g., Iterative DPO, Step DPO.

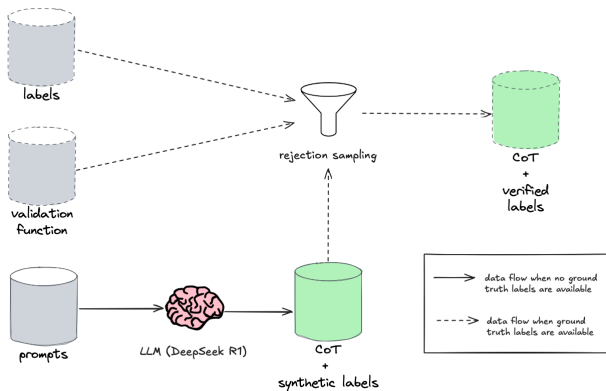
Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Human Annotation and LLM Automation



Reject Sampling Fine-tuning



Criteria:

- Unsupervised:
 - Semantic Entropy
 - Confidence
- Supervised:
 - Ground Truth (math)
 - Test Cases (code)
 - Compiler (formal math)
- Reward Model

$$\max_{\theta} J_{\text{RFT}}(\theta) = \mathbb{E}_{x \sim D} \sum_a \pi_{\theta_{\text{old}}}(a|x) \cdot \mathbf{1}[a \in H(x)] \cdot \log \pi_{\theta}(a|x) \quad (1)$$

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

The GRPO Objective Function

For each question q , GRPO samples a group of outputs $\{o_1, o_2, \dots, o_G\}$ and maximizes the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)] \\ \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL}[\pi_{\theta} || \pi_{ref}] \right\}$$

- $\hat{A}_i = R(o_i) - \frac{1}{G} \sum_{j=1}^G R(o_j)$ is the advantage calculated based on **relative rewards** within the group, not a learned value function.
- The KL-divergence penalty with a reference policy π_{ref} is added directly to the loss to stabilize training, rather than being part of the reward signal.

Algorithm Implementation

What are the probabilities, states, actions, and rewards in LLMs (consider token-wise things)? How to implement them?

Example:

```
1 def compute_reward(r: Union[torch.Tensor, float], kl_coef: float, kl: torch.Tensor, action_mask:
   Optional[torch.Tensor] = None, ) -> torch.Tensor:
2     # Step 1: Calculate the KL penalty term for each step (corresponds to  $-\beta * KL_t$ ).
3     kl_reward = -kl_coef * kl
4     # Step 2: Construct the sparse final reward. The goal is to create a tensor where the total
       sequence reward 'r' is placed only at the position of the last valid token.
5     eos_indices = action_mask.size(1) - 1 - \
6         action_mask.long().flipr().argmax(dim=1, keepdim=True)
7     last_reward = torch.zeros_like(kl)
8     last_reward.scatter_(dim=1, index=eos_indices, src=r.unsqueeze(1).to(kl.dtype))
9     # Step 3: Add the KL penalty and the final reward to get the final token-level reward r_t.
10    reward = last_reward + kl_reward
11    return reward
```

Please refer to the source code of OpenRLHF for more details.

Engineering Challenges

Packages: Ray, DeepSpeed, vllm ...

System Complexity

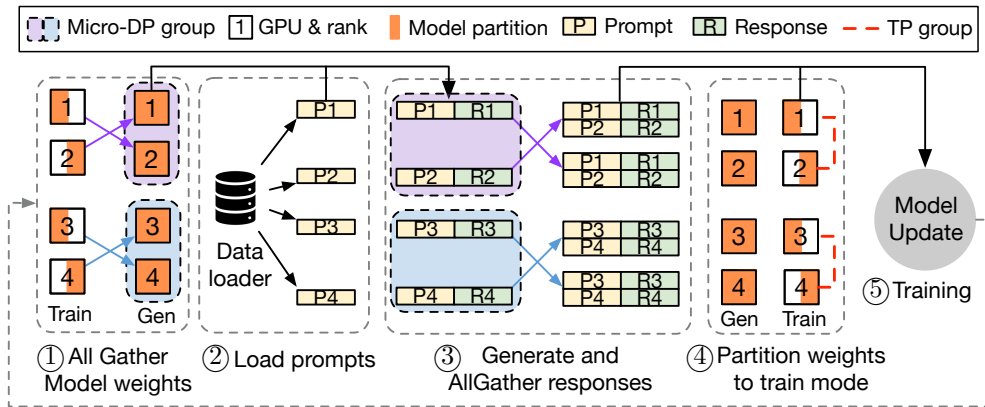


Figure: 3D-HybridEngine workflow in VeRL.

The RLHF Objective Function

We start with the standard RLHF objective, which is to maximize the reward while a KL-divergence penalty prevents the model from straying too far from a reference model.

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(y|x)} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y | x) || \pi_{\text{ref}}(y | x)]$$

- $r(x, y)$ is a (usually learned) reward function that scores how good a completion y is for a given prompt x .
- π_{ref} is a fixed reference policy (usually the SFT model).
- π is the policy we want to optimize.
- β is a hyperparameter that controls the strength of the KL penalty.

This objective seeks a policy π that balances getting high rewards with staying close to π_{ref} .

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

The Final DPO Loss Function

We can express the preference probability directly in terms of the policy π_θ :

$$p(y_w \succ y_l \mid x) = \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right)$$

Our goal is to find a policy π_θ that maximizes this probability over a given dataset of preferences $\mathcal{D} = \{(x, y_w, y_l)\}$. This is equivalent to minimizing the negative log-likelihood, which gives us the DPO loss function:

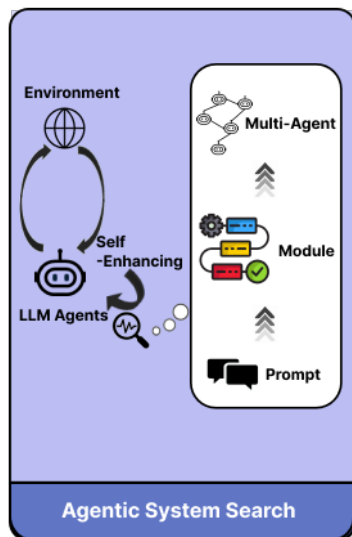
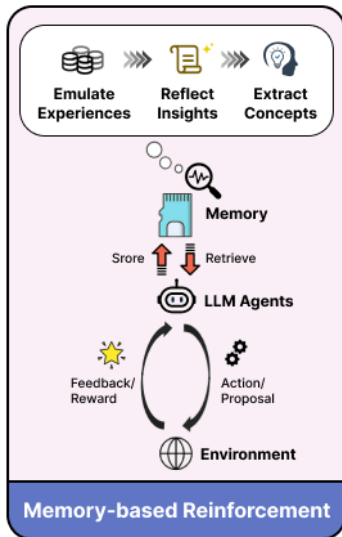
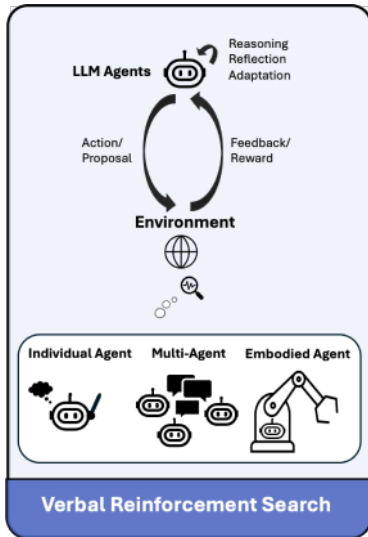
$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right]$$

This is just a simple binary cross-entropy loss that can be optimized directly with gradient descent.

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Agentic Methods



Agentic Methods + SFT

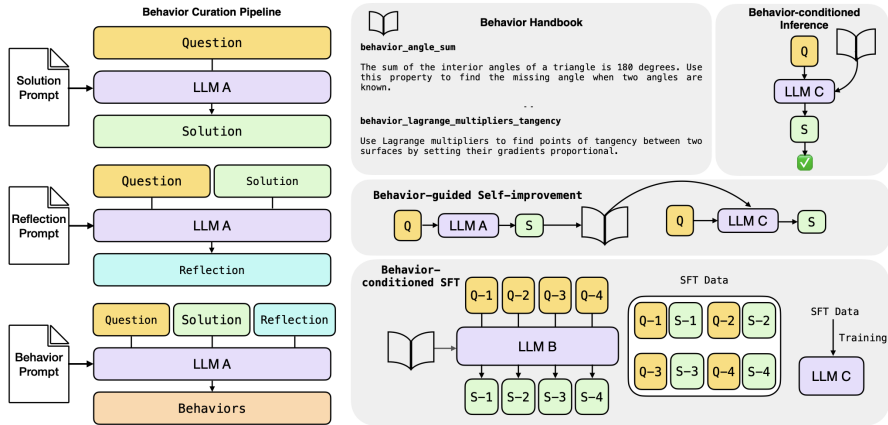


Figure: Metacognitive Reuse: Turning Recurring LLM Reasoning Into Concise Behaviors

Agentic Methods (multi agent)

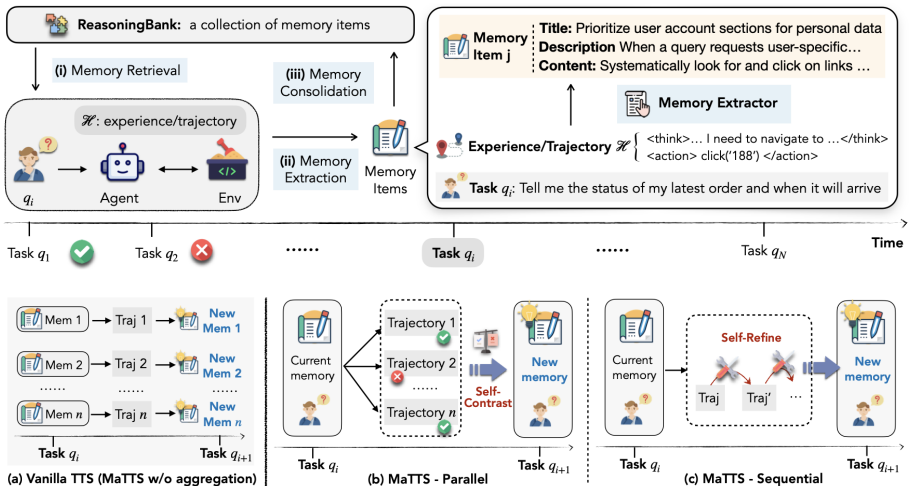
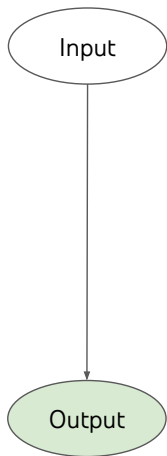
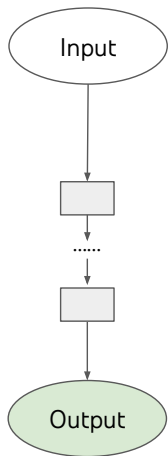


Figure: Overview of ReasoningBank, Comparison of Different Kinds of TTS

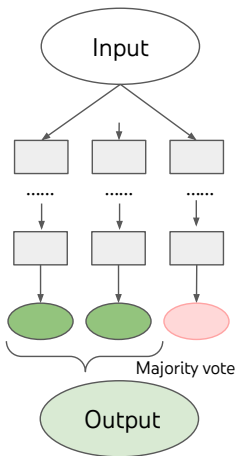
Search Methods: Tree of Thought



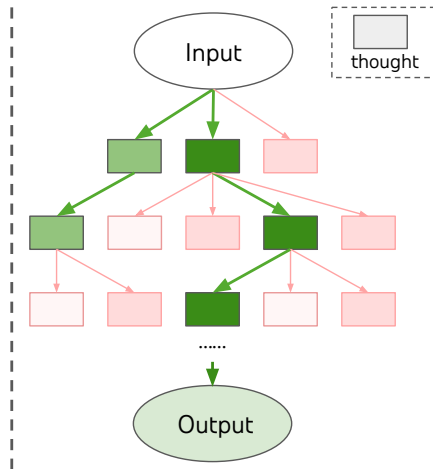
(a) Input-Output Prompting (IO)



(c) Chain of Thought Prompting (CoT)



(c) Self Consistency with CoT (CoT-SC)



(d) **Tree of Thoughts (ToT)**

Search Methods: Reasoning via Planning

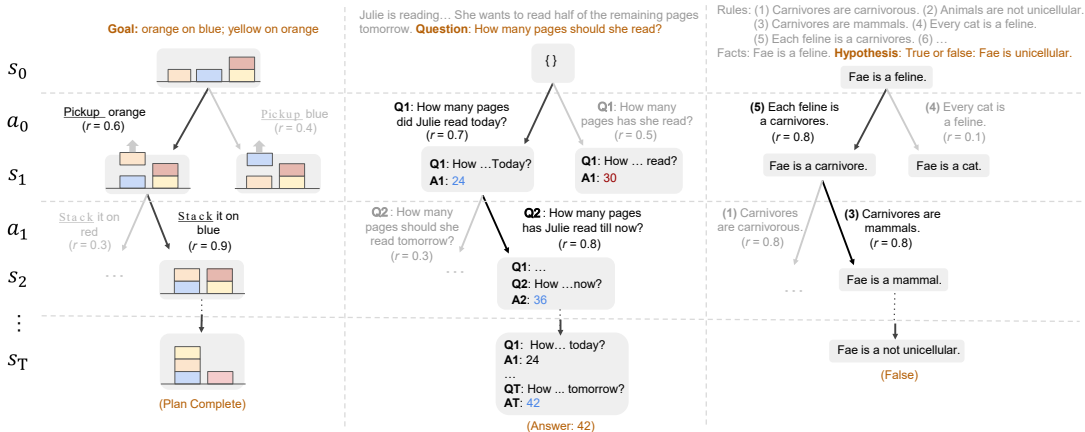


Figure: Reasoning with Language Model is Planning with World Model

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

AlphaGeometry: An Olympiad-level AI system for geometry

AlphaGeometry is a neuro-symbolic system made up of a neural language model and a symbolic deduction engine.

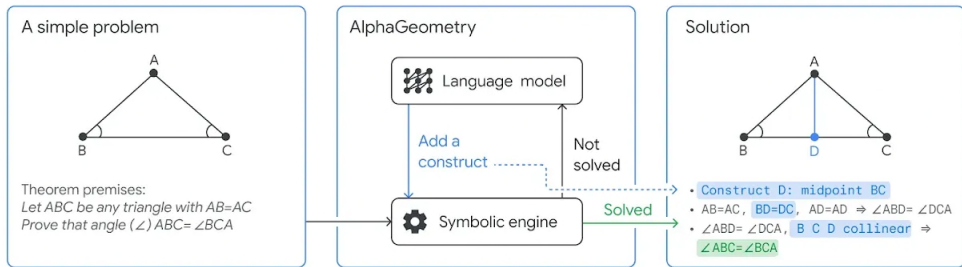


Figure: Given the problem diagram and its theorem premises (left), AlphaGeometry (middle) first uses its symbolic engine to deduce new statements about the diagram until the solution is found or new statements are exhausted. If no solution is found, AlphaGeometry's language model adds one potentially useful construct (blue), opening new paths of deduction for the symbolic engine. This loop continues until a solution is found (right). In this example, just one construct is required.

AlphaProof + AlphaGeometry 2: IMO 2024 silver medalist

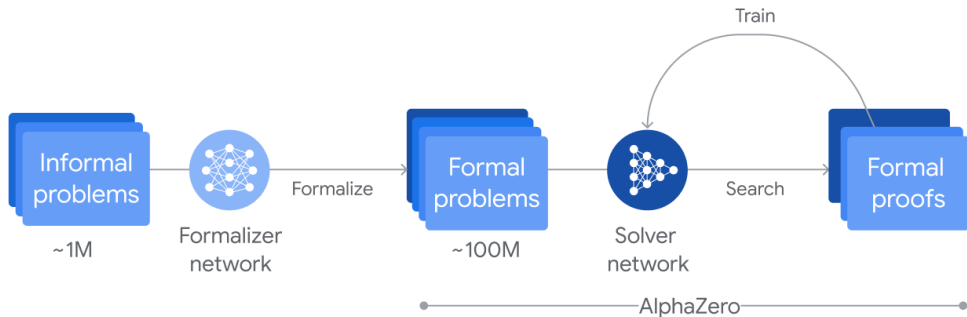


Figure: Process infographic of AlphaProof's reinforcement learning training loop: Around one million informal math problems are translated into a formal math language by a formalizer network. Then a solver network searches for proofs or disproofs of the problems, progressively training itself via the AlphaZero algorithm to solve more challenging problems.

AlphaProof is based on LEAN and Mathlib.

LEAN

In 2025, ACM SIGPLAN Programming Languages Software Award was awarded to Gabriel Ebner, Soonho Kong, Leo de Moura and Sebastian Ullrich for Lean, cited for its “significant impact on mathematics, hardware and software verification, and AI”.



Figure: A dinner with Dr. Soonho Kong (left 1). Photographed by Dr. Zhizhen Qin.

LEAN Examples (1/3): Basic Proofs

Algebraic Identity

```
1 import Mathlib
2
3 theorem a_plus_b_squared (a b : ℚ
4   ) (h1 : a - b = 4) (h2 : a *
5     b = 1) : (a + b) ^ 2 = 20 :=
6   by
7     calc
8       (a + b) ^ 2 = (a - b) ^ 2 + 4
9         * (a * b) := by ring
10        _ = 4 ^ 2 + 4 * 1 := by rw
11          [h1, h2]
12        _ = 20 := by norm_num
```

Inductive Proof

```
1 import Mathlib
2
3 theorem pow_two_ge_n_plus_one (n
4   : ℕ) : 2 ^ n ≥ n + 1 := by
5   induction n with
6   | zero => -- Base case: n = 0
7     norm_num -- Proves 1 ≥ 1
8   | succ k IH => -- Inductive step
9     calc
10       2 ^ (k + 1) = 2 * 2 ^ k :=
11         by ring
12       _ ≥ 2 * (k + 1) := by gcongr
13       _ = (k + 1 + 1) + k := by
14         ring
15       _ ≥ k + 1 + 1 := by linarith
```

LEAN Examples (2/3): A Curious Case in $\mathbb{R}$

```
1 import Mathlib
2
3 -- Disproof of the statement  $\forall x : \mathbb{R}, (x^{1/3})^3 = x$ 
4 theorem cube_root_cubed_is_false :  $\neg (\forall x : \mathbb{R}, (x^{1/3})^3 = x)$  := by
5   push_neg
6   use -1
7   -- Goal:  $((-1)^{1/3})^3 \neq -1$ 
8
9   -- Calculate  $(-1)^{1/3}$ 
10  have hcbtr :  $(-1 : \mathbb{R})^{1/3} = 1/2$  := by
11    have h_neg_one_lt_zero :  $(-1 : \mathbb{R}) < 0$  := by norm_num
12    rw [Real.rpow_def_of_neg h_neg_one_lt_zero]
13    rw [Real.log_neg_eq_log 1, Real.log_one, zero_mul, Real.exp_zero, one_mul]
14    rw [mul_comm, ←div_eq_mul_inv, Real.cos_pi_div_three]
15
16  rw [hcbtr]
17  norm_num -- Proves  $(1/2)^3 \neq -1$ 
```

The statement is false for real numbers. Lean finds a counterexample with $x = -1$. The definition of real number exponentiation for a negative base, 'Real.rpow_def_of_neg', involves the cosine of π times the exponent. This leads to $(-1)^{1/3} = \cos(\pi/3) = 1/2$. Since $(1/2)^3 \neq -1$, the general statement is disproven. This highlights the subtle pitfalls of informal reasoning.

LEAN Examples (3/3): Resolution in $\mathbb{C}$

```
1 import Mathlib
2
3 theorem complex_cube_root_cubed_eq_self (x :  $\mathbb{C}$ ) : (x ^ ((3 :  $\mathbb{N}$ )-1 :  $\mathbb{C}$ )) ^
  (3 :  $\mathbb{N}$ ) = x := by
4   have h3 : (3 :  $\mathbb{N}$ )  $\neq$  0 := by decide
5   exact Complex.cpow_nat_inv_pow x h3
```

In the complex plane, the identity holds true. The theorem 'Complex.cpow_nat_inv_pow' states that for any complex number x and any non-zero natural number n , taking the principal n -th root and then raising it to the power of n returns the original number x . The formal distinction between real and complex exponentiation is crucial, and Lean's type system enforces this rigor, preventing such errors.

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

Formal Language v.s. Informal Language

Given that $1 \leq x, y, z \leq 6$, how many cases are there in which the product of natural numbers x, y, z is divisible by 10?

Claude-3.5-Haiku Poe ...

Let's solve this systematically:

- For a product to be divisible by 10, it must have factors 2 and 5
- In the range 1 to 6, the numbers are:
 - 1 = 1
 - 2 = 2
 - 3 = 3
 - 4 = 2^2
 - 5 = 5
 - 6 = 2×3
- To be divisible by 10, the product must contain:
 - At least one 2
 - At least one 5
- Possible combinations:
 - One 5 and any other number(s)
 - One 2 and one 5
 - Multiple ways to choose these numbers from {1,2,3,4,5,6}
- Counting method:
 - Total ways to choose $x, y, z = 6^3 = 216$
 - Ways divisible by 10 = Cases with 2 and 5
- Calculating:
 - If 5 is one of x, y, z : (3 choose 1) $\times$ (5 choose 1) $\times$ (5 choose 1) = 75
 - If 2 and 5 are different numbers: (3 choose 2) $\times$ (2 choose 1) $\times$ (5 choose 1) = 60
- Total cases: $75 + 60 = 135$

The answer is 135.

Intermediate Steps ?

Final answer ✓

```
1 import Mathlib
2
3 def count_divisible_by_10 : ℕ :=
4   (finset.range 6).card * (finset.range 6).card * (finset.range 6).card
5
6 def is_divisible_by_10 (x y z : ℕ) : Prop :=
7   x * y * z % 10 = 0
8
9 theorem count_cases_divisible_by_10 :
10  let valid_numbers := (finset.range 1 7) in
11  valid_numbers.filter (λ x, valid_numbers.filter (λ y, valid_numbers.filter (λ z, is_divisible_by_10
12    x y z)).card).sum = 135 :=
13  begin
14    -- Count the valid cases
15    have h : (finset.range 1 7).prod (λ x, (finset.range 1 7).prod (λ y, (finset.range 1 7).filter (λ z,
16      is_divisible_by_10 x y z).card)) = 135,
17    { sorry }, -- Proof goes here
18    exact h,
19  end
```



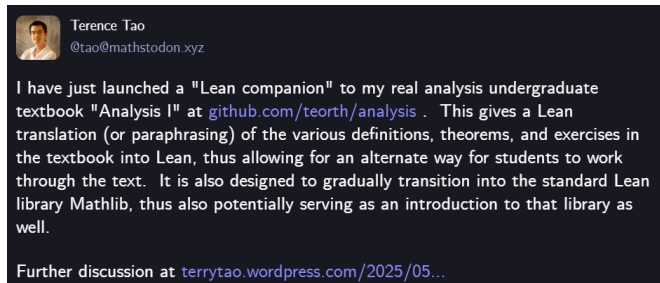
LEAN
Compiler

Machine Checkable!

Whole proof ✓

Figure: Left: Informal Language; Right: Formal Language.

The Potential of Formal Language Prover



Ya-Qin Zhang: AI can! Shing-tung Yau: AI can't!



Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

The Development of Open-Source Provers in the Past Year

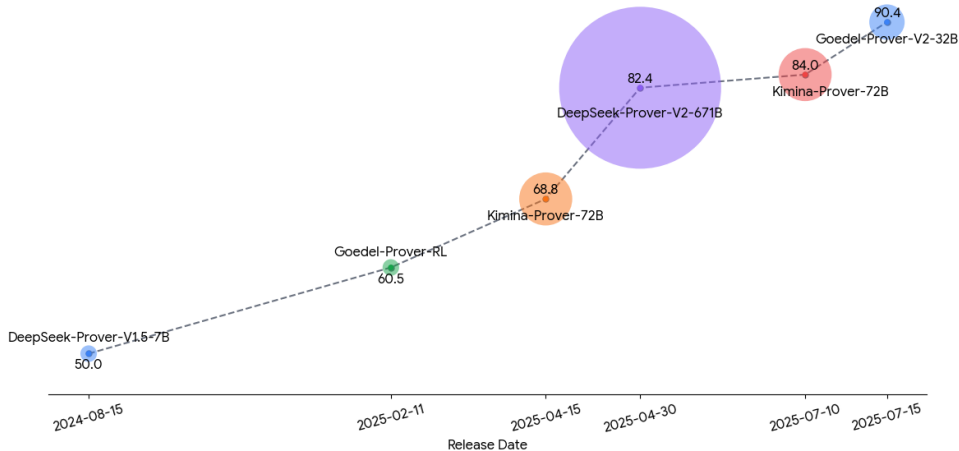


Figure: Performance of SOTA open-source provers over time (pass@32).

Achievements of Goedel-Prover-V2 - 1/2

NEW SOTA ON **MiniF2F**

+8.0% Pass@32

Our flagship 32B model significantly outperforms DeepSeek-Prover-V2-671B.

Small BUT Mighty

8B $\gtrsim$ 671B

Our small 8B model matches the prior SOTA 671B model at Pass@32.

#1 ON PutnamBench

#1

Securing the top rank on the PutnamBench leaderboard.

Achievements of Goedel-Prover-V2 - 2/2

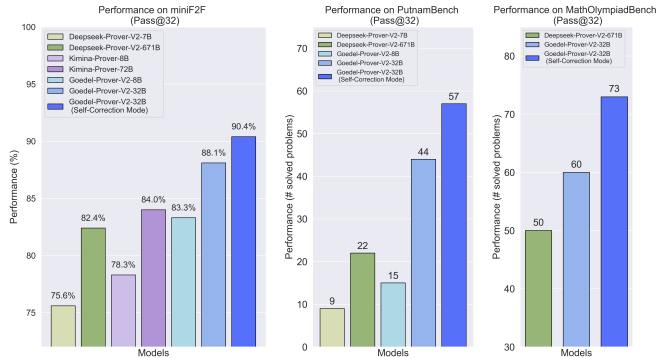


Figure: Pass@32 performance on MiniF2F, PutnamBench, and MathOlympiadBench.

Model	Solved	Pass
Goedel-Prover-V2-32B (self-correction mode)	64	Pass@64
Goedel-Prover-V2-32B (self-correction mode)	57	Pass@32
Goedel-Prover-V2-32B	44	Pass@32
DeepSeek-Prover-V2-671B	47	Pass@1024
DeepSeek-Prover-V2-671B	22	Pass@32
DSP+	23	Pass@128
Kimina-Prover-7B-Distill	10	Pass@192
Self-play Theorem Prover	8	Pass@3200
Goedel-Prover-V1	7	Pass@512

Table: PutnamBench leaderboard.
Goedel-Prover-V2-32B secures the top rank with significantly less compute.

Methods of Goedel-Prover-V2 - 1/4

Expert Iteration & RL: Formalizing problems, generating and verifying proofs, with supervised fine-tuning and reinforcement learning.

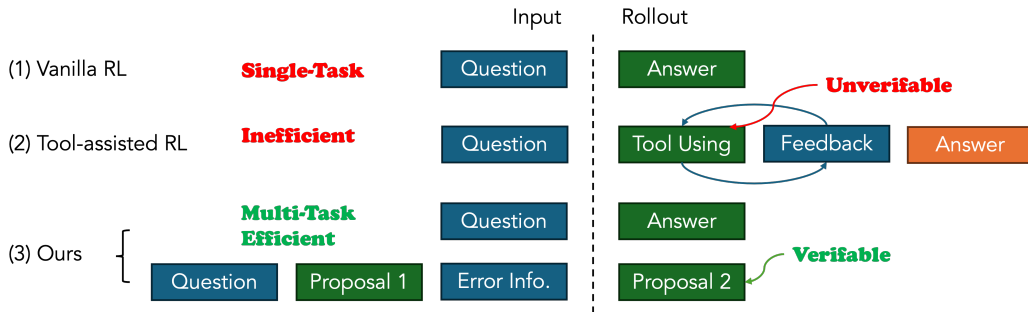


Figure: Our RL Pipeline, compared with vanilla and tool-assisted RL.

Scaffolded Data Synthesis: Implementing intermediate-difficulty problems that provide denser training signals.

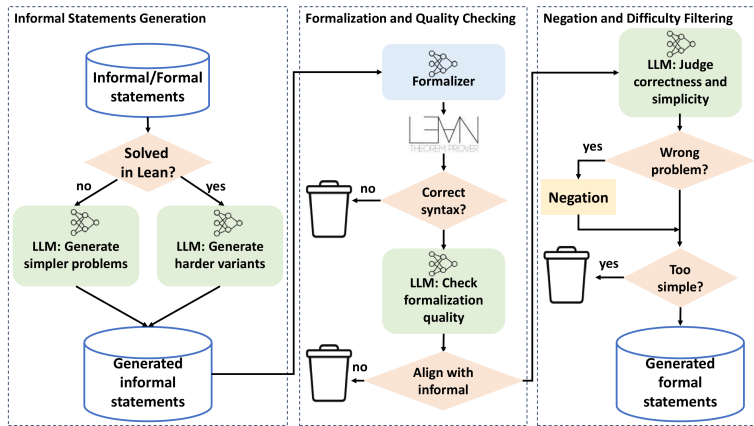
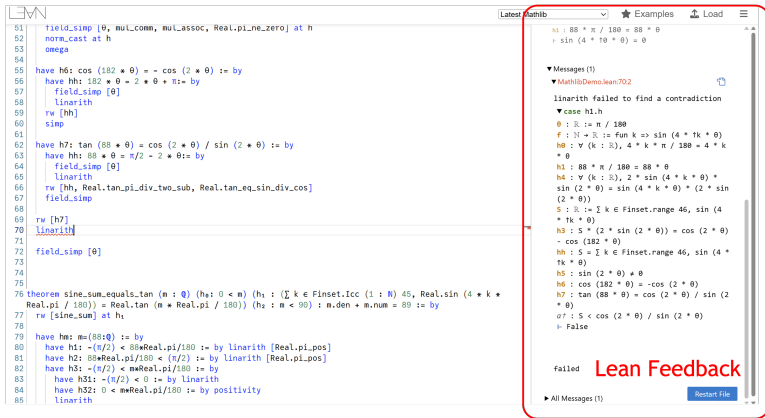


Figure: Scaffolded data synthesis pipeline.

Methods of Goedel-Prover-V2 - 3/4

Verifier-Guided Self-Correction: Our model uses Lean compilation feedback to iteratively correct its own proofs.



The screenshot displays the Lean IDE interface. On the left, a proof script is visible, starting with imports like `field_simp` and `norm_cast`, followed by several `have` statements and a `theorem` definition. The script is partially obscured by a red box on the right. This red box highlights the 'Messages' panel, which shows the output of the Lean compiler. The messages indicate that the `linarith` tactic failed to find a contradiction, and the proof is marked as 'failed'. The feedback message is displayed in red text, and a 'Restart File' button is visible at the bottom right of the messages panel.

```
51 field_simp [0, mul_comm, mul_assoc, Real.pi_ne_zero] at h
52 norm_cast at h
53 omega
54
55 have h6: cos (182 * 0) = - cos (2 * 0) := by
56   have hh: 182 * 0 = 2 * 0 + π := by
57     field_simp [0]
58     linarith
59   rw [hh]
60   simp
61
62 have h7: tan (88 * 0) = cos (2 * 0) / sin (2 * 0) := by
63   have hh: 88 * 0 = π/2 - 2 * 0 := by
64     field_simp [0]
65     linarith
66   rw [hh, Real.tan_pi_div_two_sub, Real.tan_eq_sin_div_cos]
67   field_simp
68
69 rw [h7]
70 linarith
71
72 field_simp [0]
73
74
75
76 theorem sine_sum_equals_tan (n : ℕ) (h2: 0 < n) (h1 : (∑ k ∈ Finset.Icc (1 : ℕ) 45, Real.sin (4 * k *
Real.pi / 180)) = Real.tan (n * Real.pi / 180)) (h2 : n < 90) : n.den + n.num = 89 := by
77   rw [sine_sum] at h1
78
79   have hm: m=(88:ℚ) := by
80     have h1: -(π/2) < 88*Real.pi/180 := by linarith [Real.pi_pos]
81     have h2: 88*Real.pi/180 < (π/2) := by linarith [Real.pi_pos]
82     have h3: -(π/2) < m*Real.pi/180 := by
83       have h31: -(π/2) < 0 := by linarith
84       have h32: 0 < m*Real.pi/180 := by positivity
85       linarith
```

Latest Mathlib Examples Load

h1 : 88 * π / 180 = 88 * 0
sin (4 * 0 * 0) = 0

▼ Messages (1)
▼ MathlibDemos.lean:70:2

linarith failed to find a contradiction

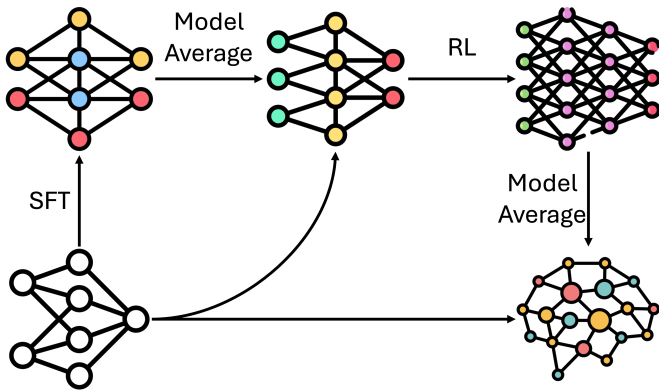
▼ case h1.h

0 : ℝ := π / 180
f : ℕ → ℝ := fun k => sin (4 * k * 0)
h0 : ∀ (k : ℝ), 4 * k * π / 180 = 4 * k * 0
h1 : 88 * π / 180 = 88 * 0
h4 : ∀ (k : ℝ), 2 * sin (4 * k * 0) * sin (2 * 0) = sin (4 * k * 0) * (2 * sin (2 * 0))
S : ℝ := ∑ k ∈ Finset.range 46, sin (4 * k * 0)
h3 : S * (2 * sin (2 * 0)) = cos (2 * 0) - cos (182 * 0)
hh : S = ∑ k ∈ Finset.range 46, sin (4 * k * 0)
h5 : sin (2 * 0) * 0
h6 : cos (182 * 0) = -cos (2 * 0)
h7 : tan (88 * 0) = cos (2 * 0) / sin (2 * 0)
a7 : S < cos (2 * 0) / sin (2 * 0)
⊢ False

failed **Lean Feedback**

► All Messages (1) Restart File

Model Averaging: Averaging trained models with the base model to restore diversity and boost Pass@K performance.



Scaling based on Goedel-Prover-V2

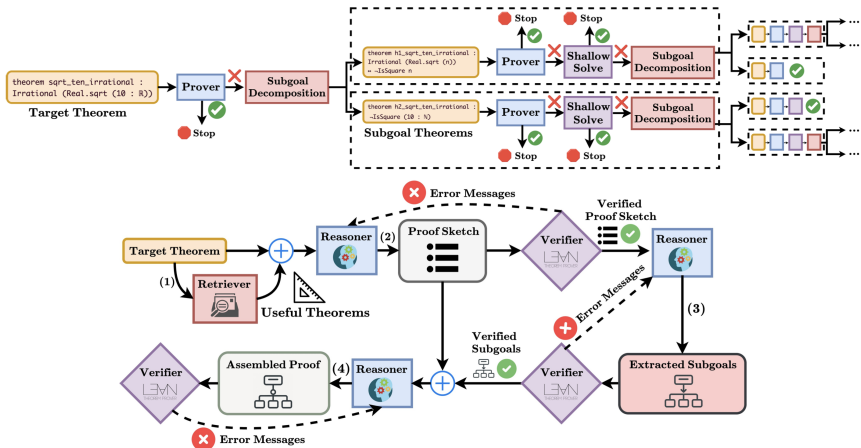


Figure: Hilbert Prover (Current SOTA)

Contents

1. Introduction of LLM Reasoning
2. Training Techniques
 - 2.1 An Overview
 - 2.2 Supervised Fine-Tuning (SFT) based Methods
 - 2.3 Group Relative Policy Optimization (GRPO)
 - 2.4 Direct Preference Optimization (DPO)
3. Inference-Time Techniques
 - 3.1 Agentic Methods
 - 3.2 Search Methods
4. Automated Theorem Proving with Formal Language
 - 4.1 Introduction to Formal Language
 - 4.2 The Potential of Formal Language Prover
 - 4.3 Development of Open-Source Provers
5. Question & Answer

- What are the main research directions, bottlenecks, and application fields of current LLMs?
- Will there be major innovations in LLM technology routes?
- How can LLMs be made to understand specific contexts (such as irony and implication)?
- How can undergraduates conduct research when lacking sufficient knowledge? How to lay the foundation for research?

Thanks!

Bohan Lyu
`bohanlyu.com`