The Curious Landscape of LLM Reasoning

An Overview of Techniques, Research Directions, and Challenges

Bohan Lyu
bohanlyu.com
Tsinghua University

October 25, 2025

Short Bio

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".

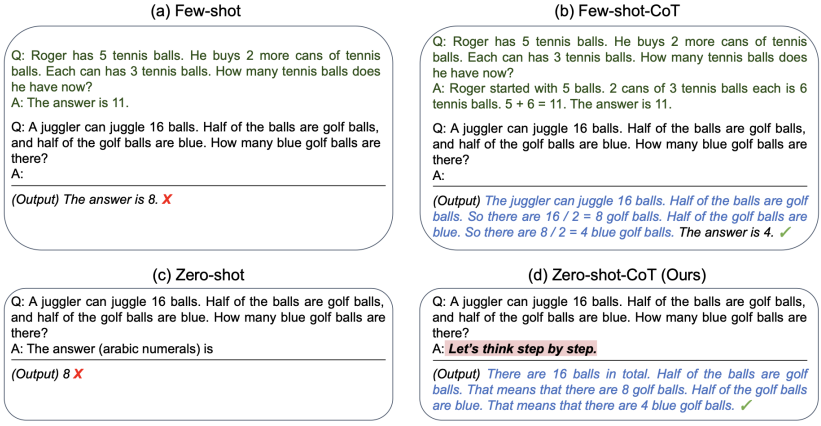


Figure: Prompting Chain of Thought.

Long CoT

OpenAI-O1, DeepSeek-R1.

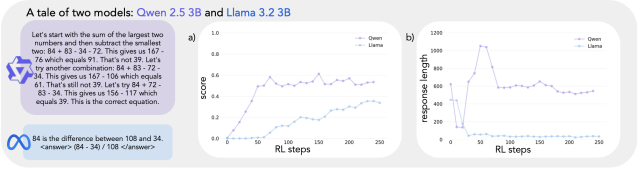
Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

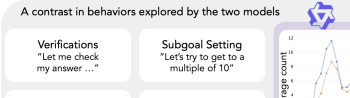
Response: <think>
To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\cdots$
$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2$.
Rearrange to isolate the inner square root term:
$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$
$\ldots$
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 $\cdots$
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: $\cdots$
$\ldots$

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.





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

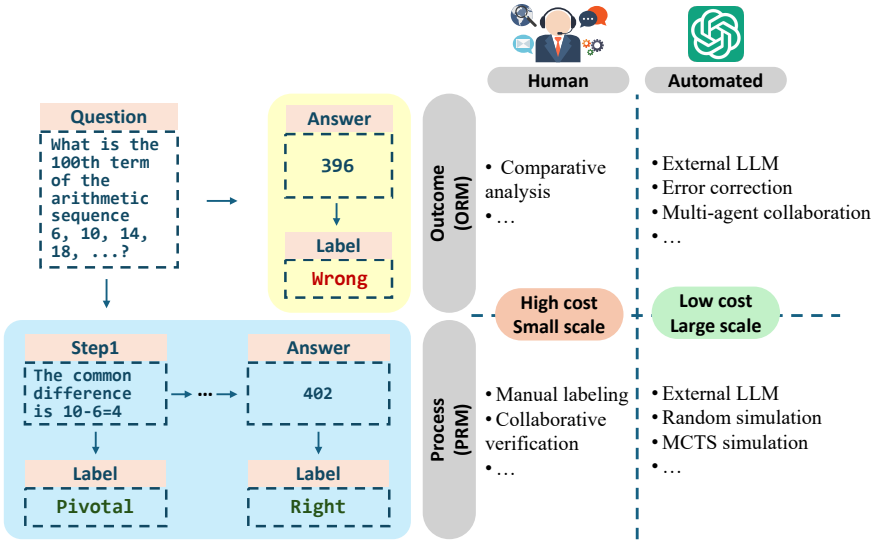
What are the techniques that can boost LLM Reasoning?

- 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.

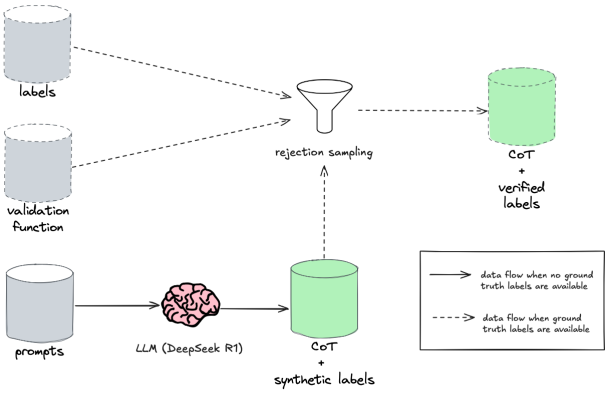
Contents

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

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

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 $\{o_1, o_2, \cdots, o_G\}$ and maximizes the following objective:

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

- $\hat{A}_i = R(o_i) - \frac{1}{G}\sum_{j=1}^{G} R(o_j)$ 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:

```python
def compute_reward(r: Union[torch.Tensor, float], kl_coef: float, kl: torch.Tensor, action_mask: Optional[torch.Tensor] = None, ) -> torch.Tensor:
    # Step 1: Calculate the KL penalty term for each step (corresponds to -beta * KL_t).
    kl_reward = -kl_coef * kl
    # 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.
    eos_indices = action_mask.size(1) - 1 - \
                  action_mask.long().fliplr().argmax(dim=1, keepdim=True)
    last_reward = torch.zeros_like(kl)
    last_reward.scatter_(dim=1, index=eos_indices, src=r.unsqueeze(1).to(kl.dtype))
    # Step 3: Add the KL penalty and the final reward to get the final token-level reward r_t.
    reward = last_reward + kl_reward
    return reward
```

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

Engineering Challenges

Packages: Ray, DeepSpeed, vllm ...
System Complexity

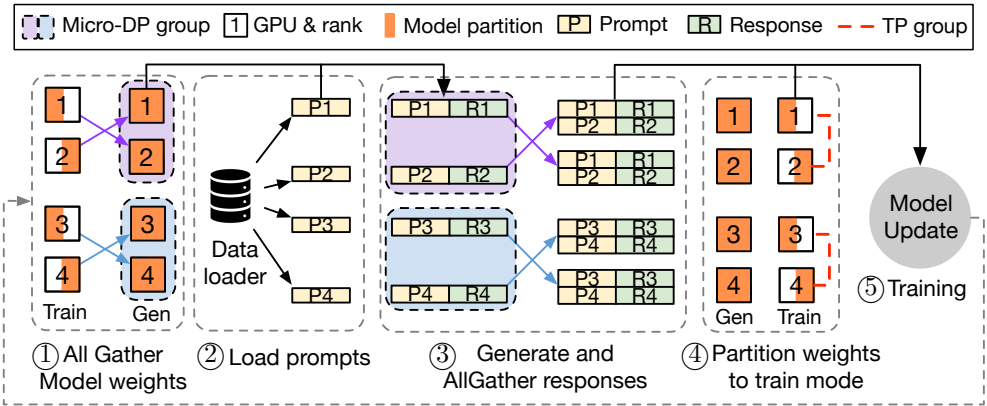


Figure: 3D-HybridEngine workflow in VeRL.

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.

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

- $r(x, y)$ 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:

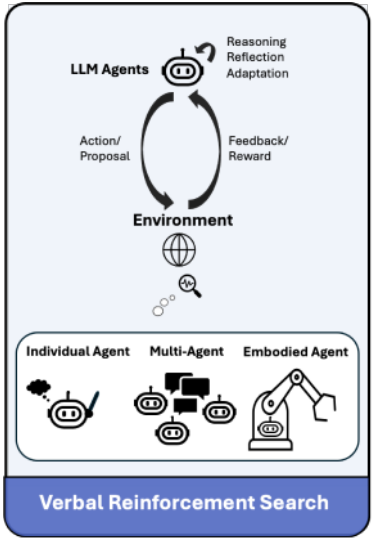
$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

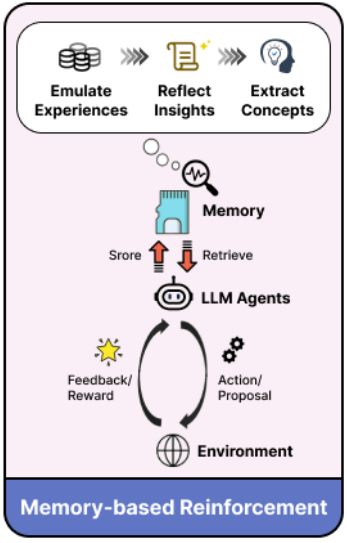
This is just a simple binary cross-entropy loss that can be optimized directly with gradient descent.

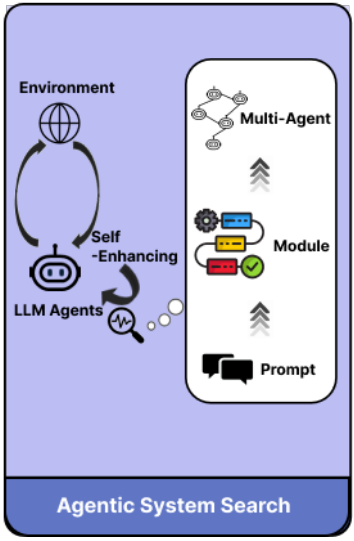
Contents

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

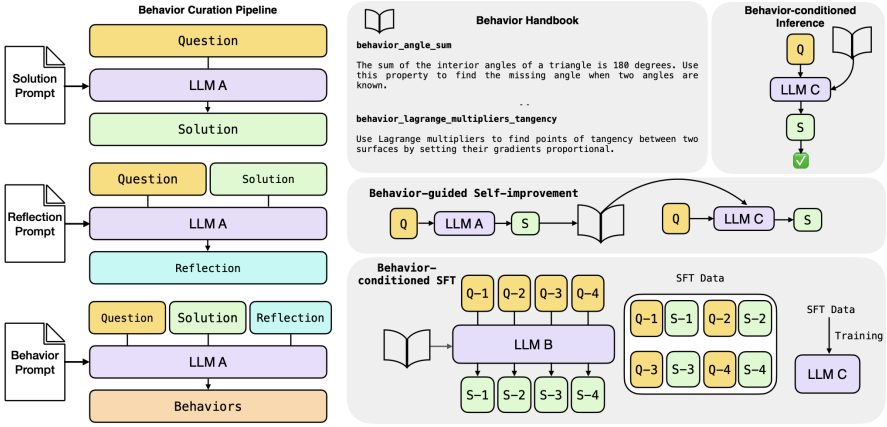


Figure: Metacognitive Reuse: Turning Recurring LLM Reasoning Into Concise Behaviors

Agentic Methods (multi agent)

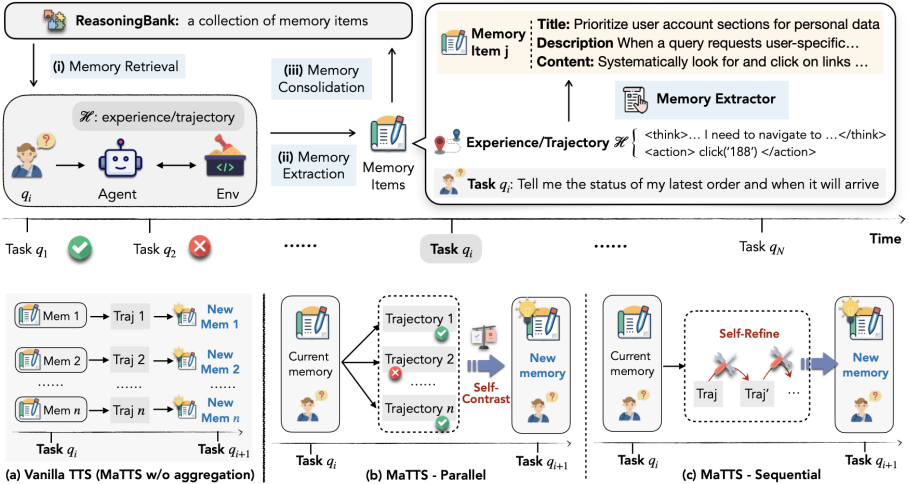
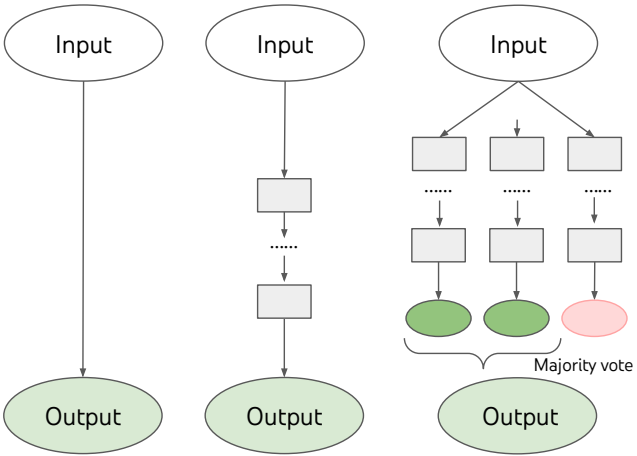


Figure: Overview of ReasoningBank, Comparison of Different Kinds of TTS

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

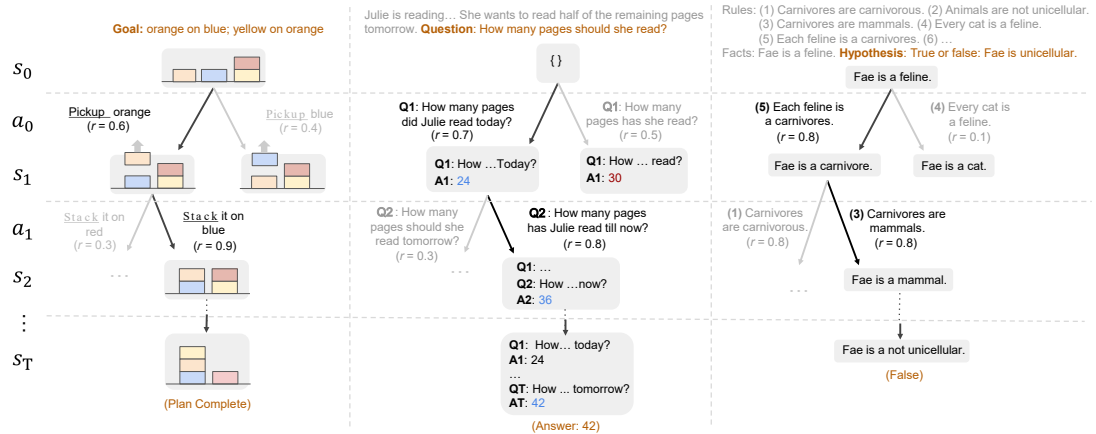


Figure: Reasoning with Language Model is Planning with World Model

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.

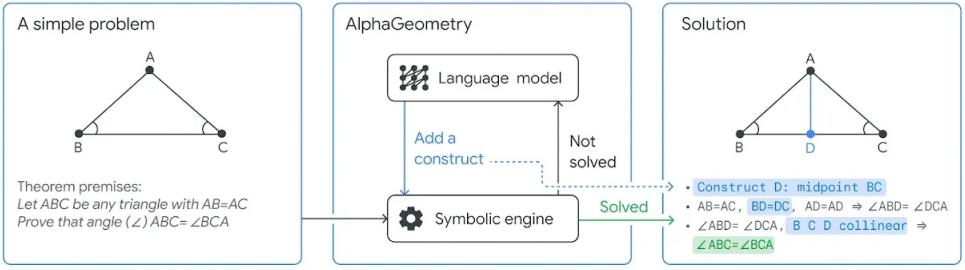


Figure: Given the problem diagram and its theorem premises (left), AlphaGeometry (middle) first uses its symbolic engine to deduce new statements about the diagram until the solution is found or new statements are exhausted. If no solution is found, AlphaGeometry's language model adds one potentially 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 madalist

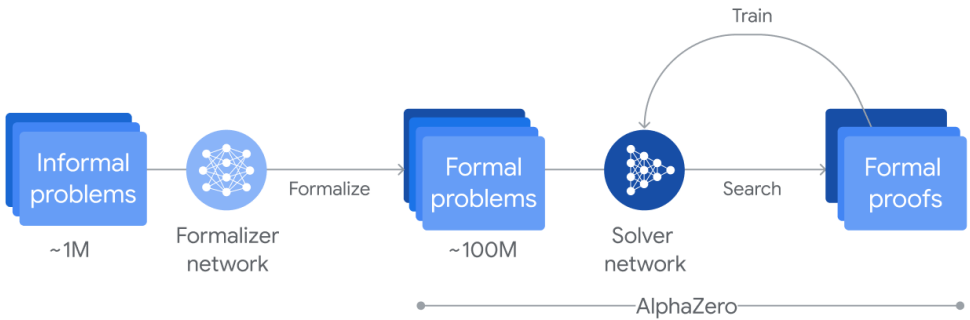


Figure: Process infographic of AlphaProof's reinforcement learning training loop: Around one million informal math problems are translated into a formal math language by a formalizer network. Then a solver network searches for proofs or disproofs of the problems, progressively training itself via the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

```
import Mathlib

theorem a_plus_b_squared (a b : ℚ
    ) (h1 : a - b = 4) (h2 : a *
    b = 1) : (a + b) ^ 2 = 20 :=
    by
  calc
    (a + b) ^ 2 = (a - b) ^ 2 + 4
    * (a * b) := by ring
    _ = 4 ^ 2 + 4 * 1 := by rw
    [h1, h2]
    _ = 20 := by norm_num
```

Inductive Proof

```
import Mathlib

theorem pow_two_ge_n_plus_one (n
    : ℕ) : 2 ^ n ≥ n + 1 := by
  induction n with
  | zero => -- Base case: n = 0
    norm_num -- Proves 1 ≥ 1
  | succ k IH => -- Inductive step
    calc
      2 ^ (k + 1) = 2 * 2 ^ k :=
    by ring
      _ ≥ 2 * (k + 1) := by gcongr
      _ = (k + 1 + 1) + k := by
    ring
      _ ≥ k + 1 + 1 := by linarith
```

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

```
import Mathlib

-- Disproof of the statement ∀ x : ℝ, (x ^ (1/3 : ℝ)) ^ (3 : ℝ) = x
theorem cube_root_cubed_is_false : ¬ (∀ x : ℝ, (x ^ (1/3 : ℝ)) ^ (3 : ℝ) = x) := by
  push_neg
  use -1
  -- Goal: ((-1 : ℝ) ^ (1 / 3 : ℝ)) ^ (3 : ℝ) ≠ -1

  -- Calculate (-1)^(1/3)
  have hcbrt : (-1 : ℝ) ^ (1 / 3 : ℝ) = 1 / 2 := by
    have h_neg_one_lt_zero : (-1 : ℝ) < 0 := by norm_num
    rw [Real.rpow_def_of_neg h_neg_one_lt_zero]
    rw [Real.log_neg_eq_log 1, Real.log_one, zero_mul, Real.exp_zero, one_mul]
    rw [mul_comm, ←div_eq_mul_inv, Real.cos_pi_div_three]

  rw [hcbrt]
  norm_num -- Proves (1/2)^3 ≠ -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 $\mathbb{C}$

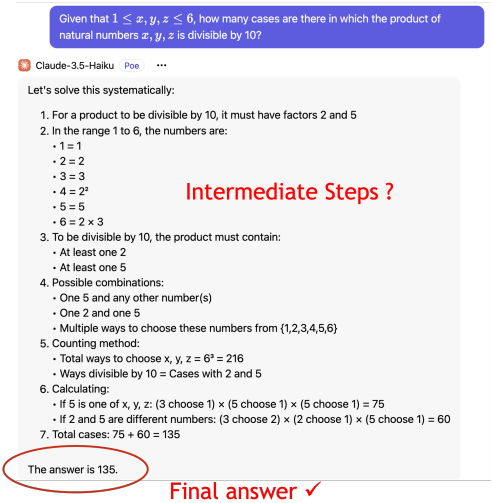
```
import Mathlib

theorem complex_cube_root_cubed_eq_self (x : ℂ) : (x ^ ((3 : ℕ)⁻¹ : ℂ)) ^
    (3 : ℕ) = x := by
  have h3 : (3 : ℕ) ≠ 0 := by decide
  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.

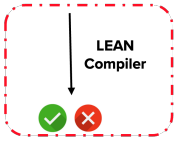
Contents

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




```
import Mathlib

def count_divisible_by_10 : ℕ :=
  (finset.range 6).card * (finset.range 6).card * (finset.range 6).card

def is_divisible_by_10 (x y z : ℕ) : Prop :=
  x * y * z % 10 = 0

theorem count_cases_divisible_by_10 :
  let valid_numbers := (finset.range 1 7) in
  valid_numbers.filter (λ x, valid_numbers.filter (λ y, valid_numbers.filter (λ z, is_divisible_by_10
  x y z)).card).sum = 135 :=
begin
  -- Count the valid cases
  have h : (finset.range 1 7).prod (λ x, (finset.range 1 7).prod (λ y, (finset.range 1 7).filter (λ z,
  is_divisible_by_10 x y z).card)) = 135,
  { sorry }, -- Proof goes here
  exact h,
end
```

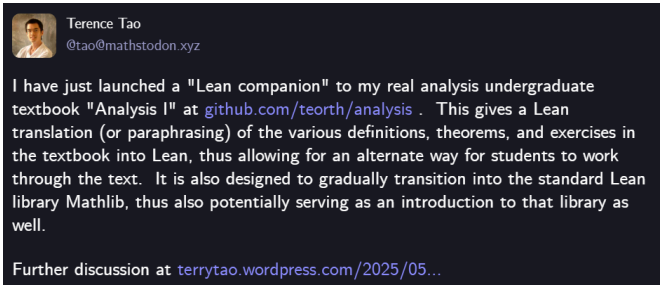


Machine Checkable!

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

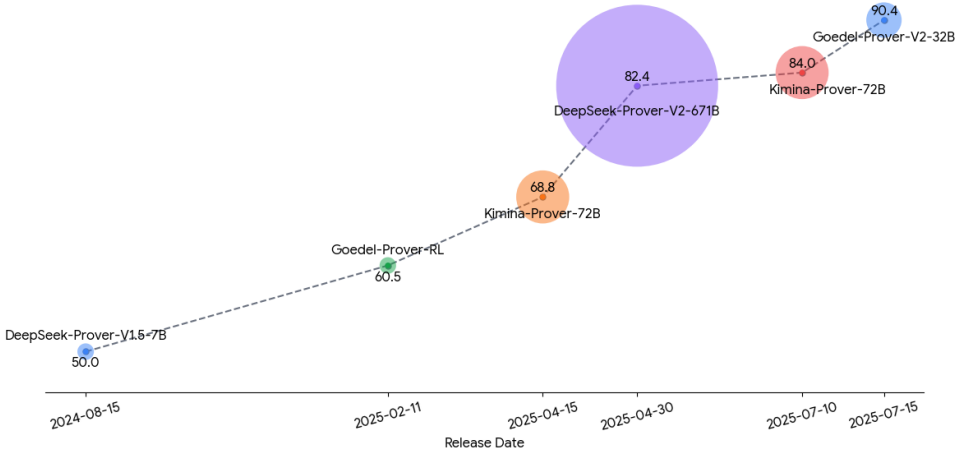


Figure: Performance of SOTA open-source provers over time (pass@32).

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

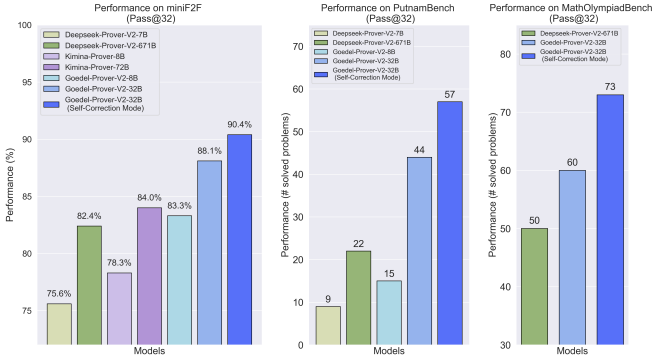


Figure: Pass@32 performance on MiniF2F, PutnamBench, and MathOlympiadBench.

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.

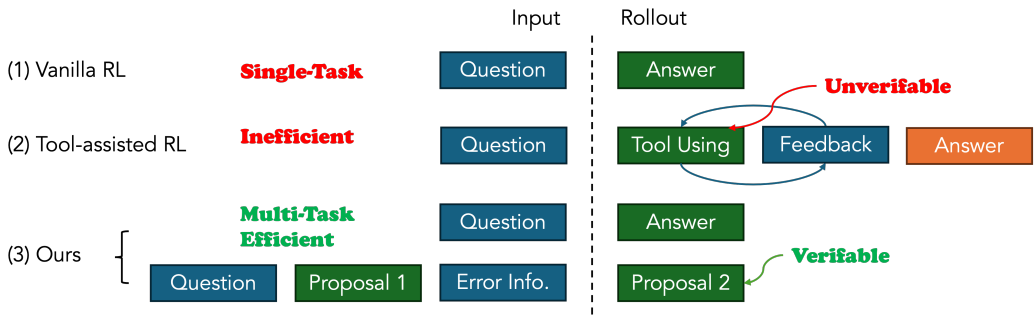


Figure: Our RL Pipeline, compared with vanilla and tool-assisted RL.

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

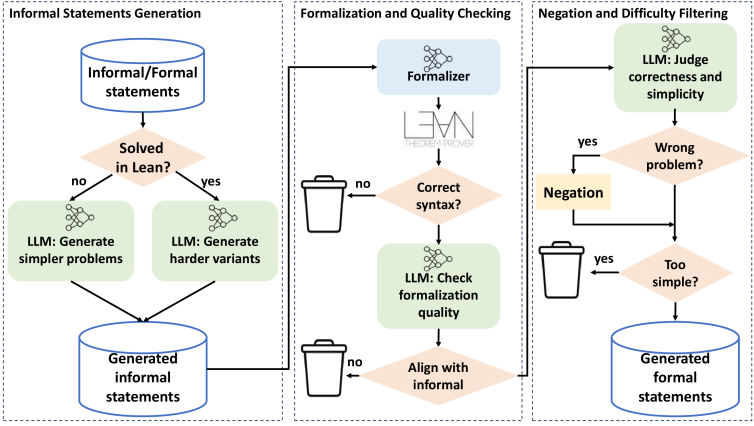
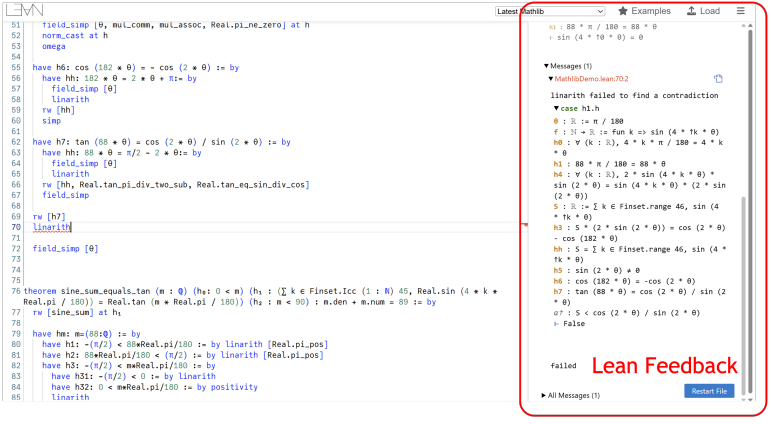


Figure: Scaffolded data synthesis pipeline.

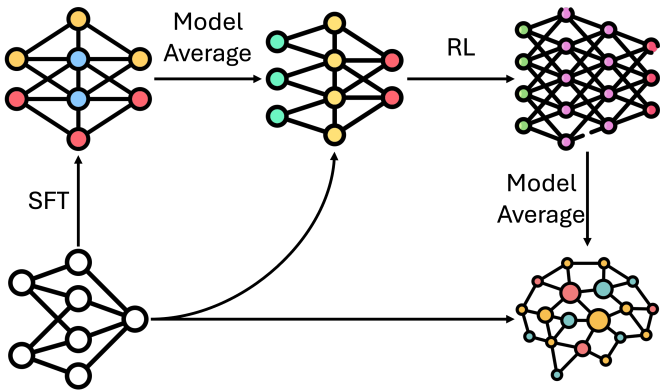
Methods of Goedel-Prover-V2 - 3/4

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



Methods of Goedel-Prover-V2 - 4/4

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



Scaling based on Goedel-Prover-V2

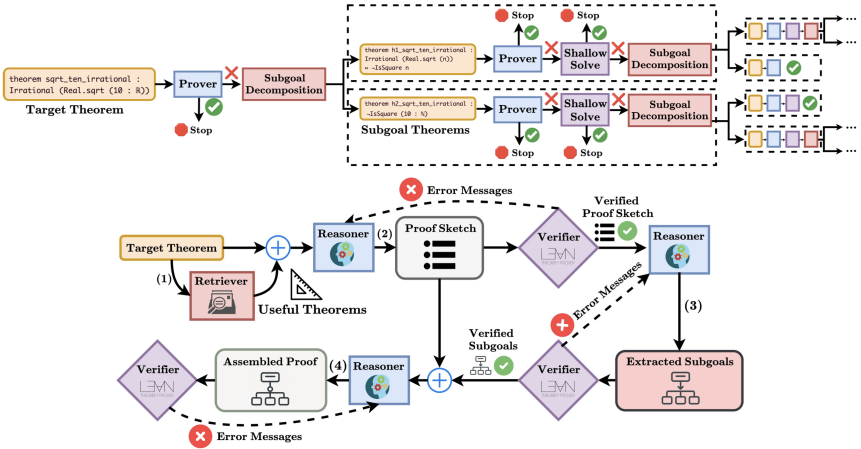


Figure: Hilbert Prover (Current SOTA)

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

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