R-PRM: Reasoning-Driven Process Reward Modeling

Anonymous ACL submission

Abstract

Large language models (LLMs) inevitably make mistakes when performing step-by-step mathematical reasoning. Process Reward Models (PRMs) have emerged as a promising solution by evaluating each reasoning steps. However, existing PRMs typically output evaluation scores directly, limiting both learning efficiency and evaluation accuracy that are further exacerbated by scarcity of annotated data. To address these issues, we propose Reasoning-Driven Process Reward Modeling (**R-PRM**). First, we leverage stronger LLMs to generate seed data from limited annotations, effectively bootstrapping our model's reasoning capabilities and enabling comprehensive step-by-step evaluation. Second, we further enhance performance through preference optimization, without requiring additional annotated data. Third, we introduce inference-time scaling to fully harness the model's reasoning potential. Extensive experiments demonstrate R-PRM's effectiveness: on ProcessBench and PRMBench, it surpasses strong baselines by 11.9 and 8.5 F1 scores respectively. When applied to guide mathematical reasoning, R-PRM achieves consistent accuracy improvements of over 8.5 points across six challenging datasets. Further analysis reveals that R-PRM exhibits better evaluation comprehensiveness and generalization capabilities, with providing additional performance gains and underscoring its potential.

1 Introduction

Recently, large language models (LLMs) have demonstrated significant progress in solving challenging mathematical problems through chain-of-thought reasoning (Wei et al., 2023; Yang et al., 2024; Shao et al., 2024). However, LLMs still inevitably make errors during the reasoning process, which reduces the reliability of their solution and prevents them from reaching correct answers.

Therefore, Process Reward Model (PRM) has been proposed to further improve models reasoning ability (Lightman et al., 2023). Unlike Outcome Reward Models (ORM) that only focus on the final results, PRM evaluate each reasoning step in a more fine-grained manner and achieve better performance and generalization, such as identifying and mitigating process errors, while also exhibiting stronger generalization capabilities (Lightman et al., 2023; Wang et al., 2024b).

A primary challenge in PRM development arises from data scarcity. While human annotation can provide high-quality process-level labels (Lightman et al., 2023), it incurs substantial costs. Alternative automated approaches, such as Monte Carlo (MC) methods that estimate step correctness based on probability of reaching correct final answer (Wang et al., 2024b,a; Luo et al., 2024b), or using stronger language models as judges for data filtering (Zhang et al., 2025), have shown some promise. However, these methods either require significant computational resources or still struggle with noise and bias, leaving the challenge of sufficient high-quality training data unresolved.

Moreover, existing process reward models directly provide the evaluation judgment based on the given steps. We argue that for challenging process-level evaluation tasks, this direct evaluation approach constrains the model's learning process and reduces learning efficiency. Meanwhile it lacks interpretability as it fails to identify why specific steps are incorrect which makes it difficult to provide constructive feedback for improvement.

To address these issues, we propose a Reasoning-Driven Process Reward Modeling (**R-PRM**) framework, which ultilizes reasoning for each intermediate steps to further enhance the PRM. The framework consists of three key components: First, we construct seed data by prompting stronger LLMs using a small amount of human-annotated step-level labels and subsequently fine-tune Qwen2.5-Math-7B-Instruct. Through this reasoning-centric paradigm, our model develops the capability to

perform comprehensive and transparent analyses for evaluating complex solution steps of challenging questions. Second, the generative evaluation paradigm enables us to apply preference optimization, improving model capabilities by encouraging the generation of evaluation processes that lead to correct judgments, without requiring additional data generation. Finally, we exploit the generative nature of our evaluation paradigm at inference time, allowing multiple evaluation processes to be sampled for a more comprehensive and robust assessment perspective.

When evaluated on ProcessBench and PRM-Bench, our R-PRM achieves F1-score improvements of 11.9 and 8.5 respectively over the strongest baseline trained on the same data. Furthermore, when deployed to guide policy model reasoning through Best-of-N and Guided Search strategies, our approach elevates accuracy by average margins of +8.6 and +8.4 over Pass@1 baselines across six challenging math datasets, outperforming both majority voting and all PRM baselines. Further analysis reveals our three key additional advantages: (1) comprehensive evaluation coverage through multi-dimension analysis, (2) enhanced generalization capability across diverse dataset, and (3) progressive accuracy improvement with increased reasoning budgets, suggesting significant potential for practical reasoning-system optimization.

2 Related Work

2.1 Mathematical Reasoning

Recent studies have demonstrated that LLMs exhibit enhanced reasoning capabilities when generating step-by-step solution before providing final answers (Wei et al., 2023). Building on this insight, several pioneering works have focused on developing large-scale mathematical datasets with high-quality reasoning annotations for fine-tuning of LLMs (Luo et al., 2025; Wang et al., 2023; Shao et al., 2024; Yang et al., 2024). However, even when models arrive at correct final answers, their intermediate reasoning steps may contain critical errors. This discrepancy undermines the reliability of their problem-solving processes and poses significant obstacles for future model improvements (Zheng et al., 2024).

Parallel advancements (Snell et al., 2024; O1, 2023; DeepSeek-AI, 2025; QwQ, 2023) in inference-time have demonstrated that increasing the computational budget to enable multiple reasoning attempts, coupled with majority voting mechanisms for answer selection, can achieve remarkable accuracy improvements.

2.2 Reward Modeling of Reasoning

Reward models are introduced to further improve mathematical reasoning by enhancing training data quality, guiding model learning(Lightman et al., 2023; Cobbe et al., 2021; Uesato et al., 2022), and guiding the policy model's reasoning process through Best-of-N and Guided-Search methods(Wang et al., 2024b; Zhang et al., 2025).

Currently, reward models are mainly divided into Outcome Reward Model (ORM) and Process Reward Model (PRM) (Lightman et al., 2023). ORM focuses on giving an overall evaluation based on whether the correct answer can be obtained ultimately (Cobbe et al., 2021). In contrast, PRM provides a fine-grained evaluation for each reasoning step, and many work has shown that PRM can achieve better results (Lightman et al., 2023; Uesato et al., 2022). However, data for PRM is extremely scarce, and the annotation cost is high (Lightman et al., 2023; Wang et al., 2024b; Luo et al., 2024b). Some research explores automatic synthesis strategies, such as using the Monte Carlo (MC) estimation method (Wang et al., 2024b; Luo et al., 2024b). However, MC requires great computation cost and inevitably introduce bias and noise (Zheng et al., 2024). (Zhang et al., 2025) propose combining MC with LLM-as-a-judge, helping to reducing noise. However, the quality and quantity of step - level reasoning evaluation data are still limited, and this remains an unsolved challenge.

3 Method

3.1 Reasoning for Process Reward Modeling

Given a mathematical problem Q, the policy model generates a sequential chain-of-reasoning process $S = \{s_1, s_2, ..., s_n\}$, where each reasoning step s_i is generated conditioned on both Q and the preceding steps $\{s_1, ..., s_{i-1}\}$. To evaluate the quality of each reasoning step, current process-level reward models employ a direct prediction mechanism that assigns a score to each step. This evaluation process can be formally expressed as:

$$R_i = M(Q, s_1, ..., s_i)$$

where $M(\cdot)$ represents the reward model that outputs a scalar reward R_i.

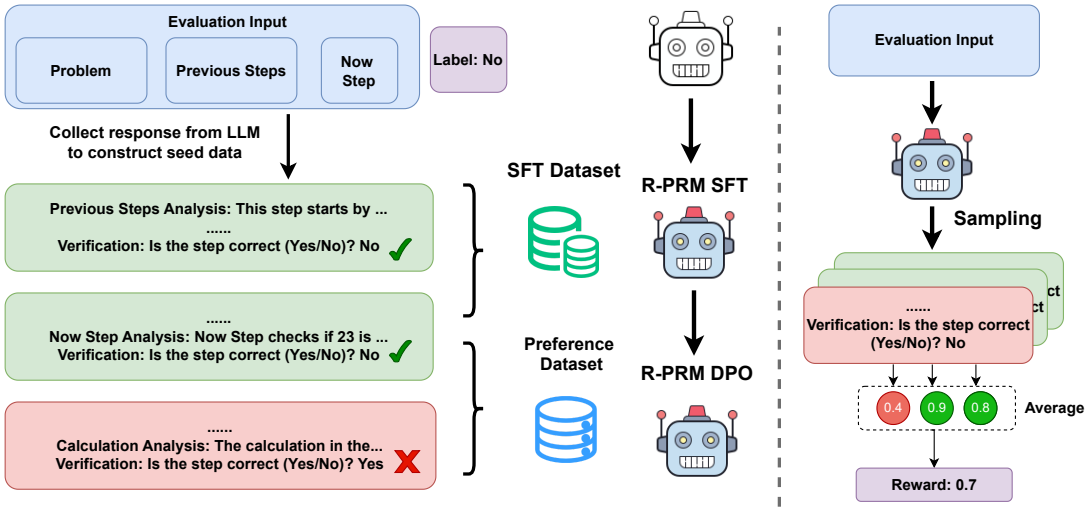

Figure 1: Illustration of our framework. For brevity, only partial analytical reasoning trajectories are shown. White robots indicate initial models, while colored ones represent models after our training procedure.

Different from previous approaches, we propose a reasoning-driven process-level reward model G that operates in a single generation sequence to perform two phases. First, PRM generates a comprehensive analysis A_i of each reasoning step s_i, consisting multiple analytical dimensions: examining historical reasoning steps, assessing the objective and data sources of the current step, verifying its coherence with preceding steps, and validating the computational transformations therein. Then, followed by analysis, model make the judgment J_i which is is a natural language phrase indicating the correctness of a result, expressed as "yes" or "no":

$$A_i = G(Q, s_1, ..., s_i)$$

$$J_i = G(Q, s_1, ..., s_i, A_i)$$

We prompt a stronger LLM with samples from PRM800K to generate (Q, $s_{1:i}$, A_i, J_i) tuples [1]. We only keep those evaluation reasoning thoughts that reach the consistent judgment with human label. Subsequently, we concatenate the analysis and score as the target sequence. Let Y_i denote the complete output sequence for s_i:

$$Y_i = A_i \oplus J_i = \{y_1, y_2, ..., y_t\}$$

$$\mathcal{L}_{\text{SFT}} = -\sum_{j=1}^{t} \log p(y_j | Q, s_{1:i}, y_{1:j-1})$$

where y_j represents the j-th token in the concatenated sequence, and t is the total length of this sequence. This is equivalent to standard instruction tuning where the model learns to generate both analysis and reward in a single forward pass.

[1]System prompt we used is listed in Appendix E

3.2 Process Reward Modeling Meta-Optimization

Through SFT, our model acquires the capability to leverage activated reasoning ability for evaluating complex mathematical reasoning trajectories. We can further improve models' performance by optimizing its reasoning process. More specifically, we leverage preference optimization methods like Direct Preference Optimization (DPO) (Rafailov et al., 2024) to encourage model to generate reasoning processes that yield correct judgments.

DPO involves an input pair (Y^w, Y^l), where Y^w is favored over Y^l. Correspondingly, we sampled multiple evaluation reasoning process, which can be categorized into two group based on whether the final prediction matches the annotated label. By treating the consistent trajectories as Y^w and the inconsistent ones as Y^l, we constructed multiple preference pairs. We freeze the original supervised fine-tuned LLM as the reference policy π_{ref} and optimize our model by following loss function:

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

3.3 Inference Time Scaling Strategy

Leveraging our PRM's capability to generate analytical reasoning processes, we propose a scalable inference strategy that enhances evaluation robustness through multiple reasoning trajectories. During inference, for each reasoning step s_i, we sample K independent analytical processes following:

$$(A_i^{(k)}, J_i^{(k)}) = G(Q, s_1, ..., s_i), k \in [1, K]$$

where each $A^{(k)}$ represents a distinct analytical reasoning process and $R^{(k)}$ is the corresponding judgment ("yes" or "no"). This multi-trajectory approach helps mitigate potential reasoning inconsistencies and stochastic variations inherent in large language models. To aggregate multiple evaluations, we calculate the average probability of "yes" judgments (using softmax with "no" judgments) as the reward:

$$R_i = \frac{1}{K}\sum_{k=1}^{K} P(J_i^{(k)} = \text{"yes"}|Q, s_1, ..., s_i, A_i^{(k)}).$$

4 Experiment

4.1 Experiment Settings

Tasks and Benchmarks: To validate the accuracy of our method in process reward modeling, we conduct evaluations on two challenging benchmark *ProcessBench* (Zheng et al., 2024) and *PRM-Bench* (Song et al., 2025).

- **ProcessBench** (Zheng et al., 2024) is designed to assess the ability to identify erroneous steps in mathematical reasoning processes. The dataset comprises 3,400 test cases, predominantly focusing on mathematical problems of varying levels of difficulty. Each test case includes a question accompanied by a LLM generated step-by-step solution, with the earliest incorrect step meticulously annotated by human experts. The primary task for the model is to either accurately identify the erroneous step or confirm the correctness of all steps in the solution.
- **PRMBench** (Song et al., 2025) constitutes a comprehensive benchmark for evaluating PRM, with particular emphasis on granular error diagnosis. It evaluates the capabilities of PRMs across three key dimensions: Simplicity, Soundness, and Sensitivity. These dimensions are further subdivided into nine specific aspects. A more detailed description of the nine aspects can be found in the Appendix A.

Furthermore, we validate the effectiveness of the reward model by applying it to guide the test-time scaling of the policy model Best-of-N, Greedy Guide Search, separately. Specifically, we conduct experiments across multiple benchmarks, including *MATH500* (Lightman et al., 2023), *Minerva Math* (Lewkowycz et al., 2022), *OlympiadBench* (He et al., 2024), *College Math* (Tang et al., 2024), *AIME24*, *AMC23* [2]. Following Zhang et al. (2025), we sample eight candidate steps with a temperature of 1 based on Qwen2.5-7B-Instruct.

- **Best of N:** simply selects the highest-scored response from N candidate solutions based on a PRM.
- **Greedy Guide Search:** the model produces N candidate steps at each generation step. The step with the highest score assigned by PRM is selected to continue sampling. This process is repeated iteratively until a complete response is generated.

Baselines: We selected the following strong process reward models as baselines.

- **Math-Shepherd** (Wang et al., 2024b): Automatically obtaining the probability of reaching the correct solution as step labels based on Monte Carlo Tree Search (MCTS).
- **Math-PSA** (Wang et al., 2024a): combining existing automatic annotation techniques (Luo et al., 2024a) and integrating data from Math-Shepherd and PRM800K datasets.
- **RLHFlow-DeepSeek/Mistral** (Dong et al., 2024): Similar to Math-Shepherd, but trained with iterative DPO.
- **Skywork-PRM-7B** (o1 Team, 2024): based on Qwen2.5-Math-Instruct and recently released by Skywork.
- **ReasonEval-7B** (Xia et al., 2025): Evaluates mathematical problem-solving step by step, assessing validity and redundancy.
- **Llemma-PRM800K-7B** (Sun et al., 2024): Trained exclusively on PRM800K from levels 1 to 3.
- **Qwen2.5-Math-PRM-7B** (Zhang et al., 2025): Based on Math-Shepherd, additionally employs LLM-as-Judge to perform consistency filtering and result in a 1.8M samples training dataset.

Implement details: We process each PRM800K case by prompting LLaMA3.3-70B-Instruct to generate four candidate evaluation trajectories, result in 289k SFT data and 269k DPO data. Training employs distinct learning rates (5e-6 for SFT, 5e-7 for DPO) with one epoch fine-tuning (batch size=128) on Qwen2.5-Math-7B-Instruct. We split out 20k samples for validation, retaining the checkpoint with minimal validation loss. By default we report the result with ten evaluation reasoning trajectories.

[2] Due to the large size of the Olympiad and College Math test sets, we randomly selected 200 samples from each to form the test subsets.

MODEL	GSM8K			MATH			OLYMPIADBENCH			OMNIMATH			Avg. F1
	error	correct	F1	error	correct	F1	error	correct	F1	error	correct	F1	
LLM-as-judge, Proprietary language models													
GPT-4o*	70.0	91.2	79.2	54.4	76.6	63.6	45.8	58.4	51.4	45.2	65.6	53.5	61.9
o1-mini*	88.9	97.9	**93.2**	83.5	95.1	88.9	80.2	95.6	87.2	74.8	91.7	82.4	87.9
LLM-as-judge, Open-source language models													
Llama-3.3-70B-Instruct*	72.5	96.9	82.9	43.3	83.2	59.4	31.0	94.1	46.7	28.2	90.5	43.0	58.0
Qwen2.5-Math-72B-Instruct*	49.8	96.9	65.8	36.0	94.3	52.1	19.5	97.3	32.5	19.0	96.3	31.7	45.5
Qwen2.5-72B-Instruct*	62.8	96.9	76.2	46.3	93.1	61.8	38.7	92.6	54.6	36.6	90.9	52.2	61.2
PRMs													
Math-Shepherd-7B*	32.4	91.7	47.9	18.0	82.0	29.5	15.0	71.1	24.8	14.2	73.0	23.8	31.5
Math-PSA-7B	48.3	88.1	62.4	29.5	72.7	41.9	20.7	65.8	31.5	15.4	68.9	25.2	40.3
RLHFlow-Mistral-8B*	33.8	**99.0**	50.4	21.7	72.2	33.4	8.2	43.1	13.8	9.6	45.2	15.8	28.4
RLHFlow-DeepSeek-8B*	24.2	98.4	38.8	21.4	80.0	33.8	10.1	51.0	16.9	10.9	51.9	16.9	26.6
Llemma-PRM800K-7B	36.7	71.0	48.4	39.2	47.8	43.1	33.1	25.1	28.5	35.4	31.5	33.4	38.4
Skywork-PRM-7B*	61.8	82.9	70.8	43.8	62.2	53.6	17.9	31.9	22.9	14.0	41.9	21.0	42.1
ReasonEval-7B	26.1	95.3	41.0	35.7	77.6	48.9	27.5	55.2	36.7	27.0	60.6	37.4	41.0
Qwen2.5-Math-7B-PRM800K*	53.1	95.3	68.2	48.0	90.1	62.6	35.7	**87.3**	50.7	29.8	**86.1**	44.3	58.5
Qwen2.5-Math-PRM-7B*	**72.0**	96.4	**82.4**	68.0	**90.4**	**77.6**	55.7	85.5	**67.5**	55.2	83.0	**66.3**	**73.5**
★ R-PRM-7B SFT	66.2	92.7	77.2	60.3	88.2	71.6	48.6	77.3	59.6	40.1	75.5	52.3	65.2
★ R-PRM-7B DPO	**72.0**	91.7	80.7	**71.2**	83.5	76.9	**60.2**	67.8	63.8	**55.5**	65.6	60.1	70.4

Table 1: Performance on ProcessBench. ★ represents the models we trained. Results marked with * come from Zhang et al. Bold text denotes the best results within PRM.

Model Name	Simplicity			Soundness					Sensitivity				Overall
	NR.	NCL.	Avg.	ES	SC.	DC.	CI	Avg.	PS	DR.	MS.	Avg.	
LLM-as-judge, Proprietary language models													
GPT-4o*	57.0	62.4	59.7	72.0	69.7	70.7	71.1	70.9	62.5	65.7	99.2	75.8	66.8
o1-mini*	65.6	63.7	64.6	74.5	67.7	73.8	72.3	72.1	61.8	64.8	100.0	75.5	68.8
PRMs													
Math-Shepherd-7B*	44.0	50.3	47.1	49.4	44.5	41.3	47.7	45.7	47.2	48.6	86.1	60.7	47.0
Math-PSA-7B	47.6	55.1	51.3	56.5	49.4	47.1	54.2	51.8	51.7	54.1	88.9	64.9	52.3
RLHFlow-Mistral-8B*	46.1	47.3	46.7	56.6	55.1	54.4	63.8	57.5	51.5	56.2	97.9	68.5	54.4
RLHFlow-DeepSeek-8B*	46.4	48.9	47.6	55.7	55.0	53.2	66.2	57.5	49.0	55.4	**99.8**	68.1	54.2
Llemma-PRM800k-7B*	49.3	53.4	51.4	56.4	47.1	46.7	53.3	50.9	51.0	53.5	93.6	66.0	52.0
Skywork-PRM-7B*	35.7	41.2	38.4	36.7	29.1	30.6	34.4	32.7	36.8	37.4	88.8	54.3	36.2
ReasonEval-7B*	**61.0**	50.1	55.5	62.1	65.9	61.5	66.0	63.9	55.6	58.0	99.5	71.0	60.0
Qwen2.5-Math-7B-PRM800K	48.6	47.8	48.2	62.1	59.4	58.7	68.5	62.2	52.9	64.0	**99.8**	72.2	58.3
Qwen2.5-Math-PRM-7B	49.0	55.1	52.1	71.8	67.3	66.3	**78.5**	71.0	57.6	69.1	99.7	75.5	65.5
★ R-PRM-7B SFT	52.7	**64.7**	**58.7**	70.1	62.7	63.4	69.5	66.4	**61.4**	67.4	98.3	75.7	64.9
★ R-PRM-7B DPO	52.2	58.2	55.2	**72.1**	**69.1**	**68.9**	75.0	**71.2**	61.2	**69.5**	99.1	**76.6**	**66.8**

Table 2: Performance on PRMBench. ★ represents the models we trained. Results marked with * come from Zhang et al. Bold text denotes the best results within PRM.

4.2 Experiment Results

R-PRM achieves high evaluation accuracy efficiently. As evidenced in Table 1 and Table 2, our approach achieves F1 scores of 65.2 and 64.9 on ProcessBench and PRMBench respectively, representing significant improvements over state-of-the-art baselines. Notably, when compared with Qwen2.5-Math-7B-PRM800K (the strongest PRM800K-based baseline), R-PRM shows performance gains of 6.7 and 6.6 F1 points on the respective benchmarks. The framework's effectiveness is further highlighted by its 7.2-point and 3.3-point advantages over the strong LLM LlaMA3.1-70B-Instruct and gpt-4o on ProcessBench. These results substantiate that our reasoning-driven evaluation paradigm successfully enhances model learning efficiency while maximizing the utility of human-annotated data.

The model's capabilities were significantly enhanced through DPO, achieving remarkable performance scores of 70.4 and 66.8 on two benchmarks. Notably, it surpassed Qwen2.5-Math-PRM (trained with 1.8M data points) on the PRMbench evaluation. These experimental results conclusively validate the efficacy of the DPO approach, highlighting the potential of generative evaluation paradigms to substantially improve assessment capabilities.

Setting	AIME24	AMC23	MATH	Olympiad Bench	College MATH	Minerva MATH	Avg.
pass@1	11.2	47.8	73.0	38.0	38.6	37.2	41.0
major@8	20.0	57.5	79.6	47.0	41.5	42.7	48.0
pass@8(Upper Bound)	33.3	82.5	88.8	58.5	47.5	57.7	61.4
Greedy Guide Search@8							
Math-Shepherd-7B	13.3	52.5	74.6	38.5	36.5	41.2	42.8
Math-PSA-7B	6.7	57.5	79.8	42.5	**41.0**	39.3	44.5
RLHFlow-PRM-Mistral-8B	10.0	57.5	73.4	37.5	38.0	41.2	42.9
RLHFlow-PRM-DeepSeek-8B	13.3	52.5	74.8	39.5	37.0	40.8	43.0
Llemma-PRM800K-7B	13.3	57.5	73.8	40.0	36.5	38.2	43.2
Skywork-PRM-7B	10.0	57.5	77.8	41.5	39.0	43.4	44.9
ReasonEval-7B	3.3	55.0	73.0	37.5	35.5	37.9	40.4
Qwen2.5-Math-7B-PRM800K	**23.3**	45.0	78.2	42.0	35.5	38.6	43.8
Qwen2.5-Math-PRM-7B	16.7	60.0	**81.0**	43.5	39.0	40.4	46.8
★ R-PRM-7B DPO	16.7	**70.0**	80.0	**46.5**	39.5	**43.4**	**49.4**

Table 3: The performance of PRM guided greedy search with policy model Qwen2.5-7B-Instruct.

R-PRM provides more comprehensive evaluation across multiple dimensions. In rigorous benchmarking with PRMBench, R-PRM- DPO demonstrates significant advantages over Qwen2.5-Math-7B-PRM800K. It achieves performance improvements of 7.0, 9.0, and 4.4 points across all three crucial evaluation dimensions. Surprisingly, it surpasses GPT-4's capabilities in terms of both completeness and sensitivity, establishing itself as a more comprehensive assessment paradigm.

More specifically, R-PRM's soundness superiority manifests through three dimensions: empirical validity, step consistency, and domain alignment. This structural strength directly enhances logical error detection capability through progressive contextual analysis, each reasoning step undergoes rigorous examination of preceding steps to identify inconsistencies. Meanwhile, particularly noteworthy are its prerequisite sensitivity, especially deception resistance advantage over o1-mini, highlights its robustness in detecting superficially valid yet logically flawed reasoning steps that conventional systems often overlook.

R-PRM demonstrates superior generalization capability. As shown in Table 1, all open-source PRMs listed, except for Qwen2.5-Math-PRM-7B and Skywork-PRM-7B whose training data sources are unknown, have training sets that only cover GSM8K and MATH. Among these PRMs, only Math-PSA-7B and Qwen2.5-Math-7B-PRM800K achieved an F1 score above 60 on some subsets of ProcessBench, while the performance of the others is relatively poor, especially on out-of-domain datasets such as OmniMATH and OlympiadBench. Except for Qwen2.5-Math-7B-PRM800K, which achieved an F1 score of 50.7 on OlympiadBench, the remaining models scored below 50, with most falling between 10 and 40. In contrast, R-PRM not only demonstrates excellent performance on the MATH dataset but also exceeds 60 on all out-of-domain datasets. This indicates that R-PRM has learned a reasoning pattern through training that is independent of the dataset, enabling it to perform well across datasets of varying difficulty levels.

R-PRM guides policy model to reach correct answer effectively. As shown in Table 3 and Table 4, our method achieves an average accuracy improvement of 8.4% and 8.6% over the Pass@1 baseline. It outperforms Qwen2.5-MATH-PRM by 2.6 and 1.6 points respectively, and also surpasses Majority Voting to reach SOTA performance. The experimental results directly demonstrate that our method's accurate reward evaluation at each reasoning step effectively guides the policy model to reach correct solutions. Furthermore, we believe our approach holds greater potential for integration with backtracking-enabled strategies like Monte Carlo Tree Search (MCTS) and multi-candidate strategies such as Beam Search, enabling more comprehensive utilization of our methodology.

5 Analysis

5.1 Effective Data Scaling

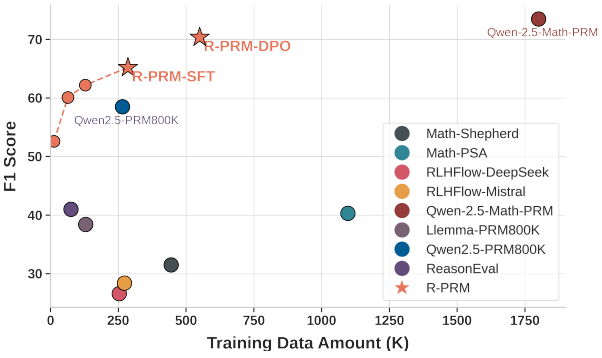

Figure 2: Average F1 score on ProcessBench with different training data scales.

Setting	AIME24	AMC23	MATH	Olympiad Bench	College Math	Minerva MATH	Avg.
pass@1	11.2	47.8	73.0	38.0	38.6	37.2	41.0
maj@8	20.0	57.5	79.6	47.0	41.5	42.7	48.0
pass@8	33.3	82.5	88.8	58.5	47.5	57.7	61.4
7B+							
Math-Shepherd-7B	16.7	42.5	76.0	42.0	37.0	39.3	42.3
Math-PSA-7B	**20.0**	55.0	80.8	47.5	39.5	40.1	47.2
RLHFlow-Mistral-8B	10.0	55.0	76.8	42.0	39.5	37.1	43.4
RLHFlow-DeepSeek-8B	13.3	57.5	76.2	40.0	39.0	39.7	44.3
Llemma-PRM800K-7B	10.0	52.5	76.6	42.5	39.0	42.7	43.9
Skywork-PRM-7B	16.7	55.0	81.2	44.0	40.5	**44.5**	47.0
ReasonEval-7B	6.7	55.0	75.2	41.0	40.0	40.4	43.1
Qwen2.5-Math-7B-PRM800K	13.3	57.5	80.0	44.5	**43.5**	43.0	47.7
Qwen2.5-Math-PRM-7B	16.7	55.0	82.0	48.0	**43.5**	43.0	**48.0**
★ R-PRM-7B DPO	**20.0**	**62.5**	**82.2**	**48.0**	41.0	44.1	**49.6**

Table 4: Performance comparison on the Best-of-8 strategy of the policy model Qwen2.5-7B-Instruct. ★ represents the models we trained.

We conduct comprehensive experiments to analyze the impact of training data scale on model performance. Figure 2 illustrates the performance vaiation of R-PRM on ProcessBench across different training data volumes. With merely 12.8k training samples, R-PRM achieves a impressive F1 score of 52.6, already surpassing most existing open-source PRMs. As we gradually increase the training data to 64k samples, R-PRM demonstrates superior performance, outperforming Qwen2.5-Math-7B-PRM800K (265k) by a margin of 1.6 points. Further scaling the training data to full training set (285k) samples yields substantial improvements, pushing the performance to 65.2.

Without requiring additional labeled data, we leverage 269K preference pairs from existing sampling results, effectively boost the model performance from 65.2 to 70.4, demonstrating exceptional data utilization efficiency.

5.2 Inference-Time-Scaling

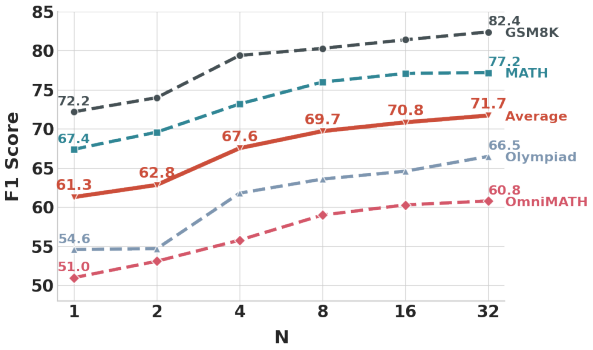

Figure 3: Efficient scaling inference-time compute on ProcessBench.

We conducted a investigation into how R-PRM's performance scales with increasing inference budgets. As shown in Figure 3, R-PRM demonstrates consistent performance gains on ProcessBench as the number of evaluation trajectories increases. A particularly significant improvement is observed when scaling from 2 to 4 samples, resulting in an increase in the ProcessBench F1 score from 62.8 to 67.6. Moreover, increasing the number of reasoning steps consistently yielded performance improvements across all four datasets, demonstrating the robustness of our scaling strategy, a unique capability inherent to our reasoning-driven approach.

5.3 Threshold Robustness in Model Evaluation

During evaluations of ProcessBench and PRMBench, we simply use 0.5 as the threshold for binary classification to determine whether the current step is correct or incorrect. We further analyze the model's sensitivity to threshold variations. As shown in Figure 4, our R-PRM demonstrates strong robustness to threshold changes, with minimal fluctuations in the ProcessBench. In contrast, Qwen2.5-Math-7B-PRM800K is more sensitive to threshold changes, exhibiting a noticeable rightward shift in its ProcessBench.

On the more challenging out-of-domain test subset OlympiadBench, our method maintains its robustness to threshold variations, while Qwen2.5-Math-7B-PRM800K exhibits a larger threshold shift, indicating its tendency to misclassify incorrect solutions as correct. This demonstrates that our method has better generalization capability, maintaining more accurate evaluation even when problem types and domains change. For detailed performance analysis of scores and thresholds on PRMBench, please refer to the Appendix B.

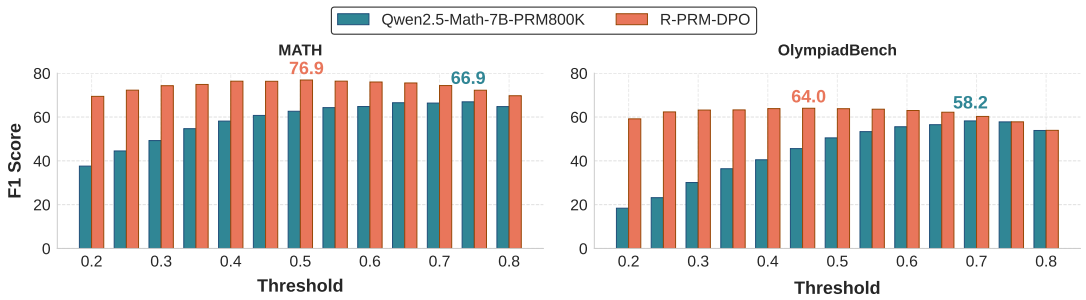

Figure 4: Average F1 score on ProcessBench at different thresholds, with the overall performance on the left and performance on the OlympiadBench dataset on the right.

Problem: An *emirp* is an integer that, when its digits are written in reverse order, is a prime number. For example, 73 is an emirp because 37 is prime. What is the smallest two-digit prime number that is NOT an emirp?

Solution: . . .
Step7: Continue to the next smallest two-digit prime number. 19 is prime.
Step 8: Check if 23 is an emirp. Reverse the digits of 23: 32. Check if 32 is prime: No, 32 is not prime (divisible by 2). Since 23 is a prime number and its reverse, 32, is not a prime number, 23 is not an emirp. . . .

Qwen2.5_Math-PRM-7B correctness score: 0.9883

Qwen2.5_Math-7B-PRM800K correctness score: 0.8594

R-PRM: Analysis: Previous Steps Analysis: . . .
. . . . . .
Calculation Analysis: . . . . . **Upon reviewing, the mistake in the analysis is assuming 23 is the answer without properly checking if there are smaller primes that are not emirps**. . . . . **Given the oversight in the analysis regarding the thorough checking of primes and their reversals, the conclusion that 23 is the smallest such prime without properly validating against all primes in sequence is premature.** . . . . . .
Conclusion: **The Now Step incorrectly concludes that 23 is the smallest two-digit prime number . . . . . .**
Verification: Is the step correct (Yes/No)? No

R-PRM (Majority Voting) score: 0.0547

Figure 5: A case study from ProcessBench MATH dataset. Red text denotes the error step and the scores of other models, and the blue text highlights our model's critique of the error and our score for that step.

5.4 Case Study

As shown in Figure 5, solution erroneously skipped the verification of number 19 in Step 7 and directly proceeded to check number 23 in Step 8. Unfortunately, both of two strong baselines, Qwen2.5-Math-PRM and Qwen2.5-Math-7B-PRM800K, fail to detect the omission, resulting in mistakenly assign high reward for Step 8 (0.99 and 0.86 respectively) . In contrast, R-PRM first performed analyses towards previous and current solution steps. Based on these analyses, R-PRM concluded that the task required verifying the numbers in ascending order, showcasing its advanced logical reasoning capability. Subsequently, R-PRM resumed the reasoning process for Step 7 to verify the correctness of number 19, thus identifying the discrepancy between its own result and the answer in Step 8. Through a comprehensive process of logical reasoning, recalculation, and verification, R-PRM assigned a reward value of 0.05 to Step 8, successfully detecting the error. Please refer to the Appendix C for more cases.

6 Conclusion

In this paper, we presented Reasoning-Driven Process Reward Modeling (R-PRM), a novel framework that advances the process reward modeling of mathematical reasoning. Our framework consists of three parts: First, we leverage stronger LLMs to construct seed data, enabling our model with comprehensive evaluation process. Second, we use preference optimization to enhance performance without requiring additional annotated data. Third, we introduce inference-time scaling to fully harness the model's reasoning capabilities. Extensive experiments demonstrate that our method achieves significant performance improvements on ProcessBench and PRMbench, while also proving effective in guiding LLM reasoning. Analysis shows that R-PRM exhibits more comprehensive, robust, and generalizable evaluation capabilities with its performance continues to improve with increased inference, highlighting the substantial potential.

Limitations

Despite extensive experiments validating our method's effectiveness on 7B models, we have not yet verified our approach on larger models such as 70B or beyond due to computational resource constraints. We hypothesis that larger models, with their enhanced reasoning capabilities, could potentially achieve higher modeling accuracy when combined with our methodology. Additionally, while we have tested popular inference strategies like Best-of-N and Guided Search, our exploration of advanced search algorithms remains limited. We have not thoroughly investigated sophisticated search methods such as Monte Carlo Tree Search (MCTS) and Beam Search, which could potentially leverage the characteristics of PRM more effectively and yield superior generation results.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. Preprint, arXiv:2110.14168.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Preprint, arXiv:2501.12948.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. Preprint, arXiv:2405.07863.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. arXiv preprint arXiv:2402.14008.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. Advances in Neural Information Processing Systems, 35:3843–3857.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. arXiv preprint arXiv:2305.20050.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. Preprint, arXiv:2308.09583.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024a. Improve mathematical reasoning in language models by automated process supervision. Preprint, arXiv:2406.06592.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024b. Improve mathematical reasoning in language models by automated process supervision. arXiv preprint arXiv:2406.06592.

OpenAI O1. 2023. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-02-08.

Skywork o1 Team. 2024. Skywork-o1 open series. https://huggingface.co/Skywork.

QwQ. 2023. Qwq: Reflect deeply on the boundaries of the unknown. https://qwenlm.github.io/blog/qwq-32b-preview//. Accessed: 2025-02-08.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. Preprint, arXiv:2305.18290.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. Preprint, arXiv:2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:2408.03314.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. Preprint, arXiv:2501.03124.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. Preprint, arXiv:2403.09472.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. arXiv preprint arXiv:2403.02884.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. Preprint, arXiv:2211.14275.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, Linyi Yang, Ying Wen, and Weinan Zhang. 2024a. Openr: An open source framework for advanced reasoning with large language models. Preprint, arXiv:2410.09671.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9426–9439.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. Preprint, arXiv:2212.10560.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. Preprint, arXiv:2201.11903.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. Preprint, arXiv:2404.05692.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. Preprint, arXiv:2409.12122.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. Preprint, arXiv:2501.07301.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. Preprint, arXiv:2412.06559.

A Detailed Description of PRMBench Subcategories

- **Non-Redundancy (NR)**: Evaluates the model's ability to identify and eliminate unnecessary steps within the reasoning process, ensuring efficiency without sacrificing correctness.
- **Non-Circular Logic (NCL)**: Assesses whether the model can detect circular reasoning, where conclusions are reintroduced as premises, leading to logical loops.
- **Empirical Soundness (ES)**: Measures the model's capability to identify and reject reasoning steps that contradict established facts or real-world knowledge.
- **Step Consistency (SC)**: Evaluates whether the reasoning steps maintain consistency with each other, ensuring that all steps logically flow from one to the next.
- **Domain Consistency (DC)**: Assesses the model's ability to apply domain-specific knowledge correctly, avoiding the misuse of concepts or theories across different domains.
- **Confidence Invariance (CI)**: Tests whether the model maintains appropriate confidence levels throughout the reasoning process, especially when errors are detected or uncertainties arise.
- **Prerequisite Sensitivity (PS)**: Evaluates whether the model detects missing prerequisites or conditions essential for valid reasoning, ensuring the completeness of the logic.
- **Deception Resistance (DR)**: Measures the model's ability to detect and reject misleading information that might appear correct but contains subtle errors.
- **Multi-Solution Consistency (MS)**: Assesses the model's ability to handle multiple valid solutions to the same problem, ensuring consistency across different reasoning paths.

B Threshold Robustness on PRMBench

As illustrated in Figure 6, the experimental results of PRMBench show that R-PRM has significant robustness advantages, while Qwen2.5-Math-7B-PRM800K exhibits a performance gap of 8.2 points between the 0.5 threshold and its optimal performance.

C Additional Case

As shown in this case, both Qwen2.5-Math-PRM-7B and Qwen2.5-Math-7B-PRM800K fail to detect the calculation error in this step, yet they show

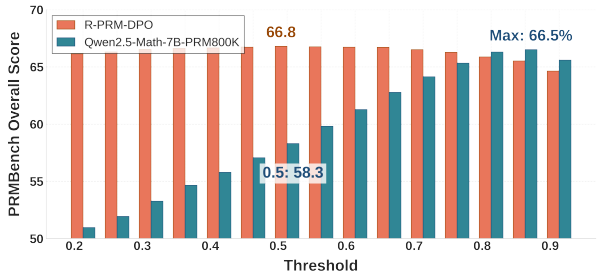

Figure 6: PRMBench Overall Score with Different Threshold

high confidence in their judgments with their correctness scores around 0.9. On the other hand, our R-PRM shows its strong ability by explaining the step, checking the calculation, and finding the error, with a final score of around 0.3.

D Used Scientific Artifacts

Below are the scientific artifacts used in our work. For the sake of ethics, our use of these artifacts is consistent with their intended use.

- *Transformers (Apache-2.0 license)*, a framework to facilitate downloading and training state-of-the-art pretrained models.
- *trl (Apache-2.0 license)*, a full stack library that provides a set of tools to train transformer language models with Reinforcement Learning. The library is built on top of the Transformers library.
- *PRM800K(MIT license)*, a large-scale dataset proposed by OpenAI, containing 800,000 step-level correctness labels for model-generated solutions to problems from the MATH dataset.
- *PRMBench(Apache-2.0 license)*, a benchmark designed to evaluate the ability of PRMs to identify error-types in the solution process, consisting of 6,216 fine-grained data instances.

E Prompt for Construction Data

Problem: Suppose t is a positive integer such that $\text{lcm}[12, t]^3 = (12t)^2$. What is the smallest possible value for t?

Solution:
To solve this problem, we can start by using the properties of the least common multiple (LCM). Given: $\text{lcm}[12, t]^3 = (12t)^2$.
Step 1: Take the cube root of both sides of the equation. $\sqrt[3]{\text{lcm}[12, t]^3} = \sqrt[3]{(12t)^2}$. This simplifies to: $\text{lcm}[12, t] = 12t^{2/3}$.
... Step 31: We can see that $13824 = 2304^2$, so $t = 48$ satisfies the original equation. Therefore, the smallest possible value for t is $\boxed{48}$.

Qwen2.5-Math-PRM-7B score: 0.9414

Qwen2.5-Math-7B-PRM800K score: 0.8711

R-PRM (Ours): Analysis: Previous Steps Analysis: Step 1: This step states the problem and sets up the equation to be solved, which is $\text{lcm}[12, t]^3 = (12t)^2$.This step is a correct restatement of the problem .....
...
Calculation Analysis: The calculation involves taking the cube root of $(12t)^2$. By the properties of exponents, $\sqrt[3]{(12t)^2} = \sqrt[3]{12^2 \cdot t^2} = 12^{2/3} \cdot t^{2/3}$. This is because the cube root of a product can be expressed as the product of the cube roots, and the cube root of a power can be expressed as that power divided by 3. Therefore, the correct simplification should be $\text{lcm}[12, t] = 12^{2/3} \cdot t^{2/3}$, not $12t^{2/3}$ as stated in the Now Step.
Conclusion: The Now Step incorrectly simplifies the expression $\sqrt[3]{(12t)^2}$ to $12t^{2/3}$. The correct simplification should be $12^{2/3} \cdot t^{2/3}$, which is different from what is presented. Therefore, the Now Step contains an error.
Verification: Is the step correct (Yes/No)? No"

R-PRM score: 0.2992

Figure 7: A case study from MATH subset of ProcessBench. Red text denotes the error step and the scores of other models, and the blue text highlights our model's critique of the error and our score for that step.

You are an excellent math teacher. Please verify the correctness of the Now Step.

You first need to analyze the Now Step and the Previous Steps and then summarize based on your analysis.
Analysis:
You need to analyze the following aspects.
Previous Steps Analysis: You need to analyze the Previous Steps step by step. For each step, you need to first explain what the current step is doing, then you try to find any error in the current step.
Now Step Analysis: You first need to explain what the Now Step is doing, and then point out which part of the Question it is trying to solve or which part of the information it states.
Data Source Analysis: First you need to find out what data are used in the Now Step, and then you need to determine whether the source of the data is reasonable and correct. When you judge whether the source of a data is reasonable and correct, you need to specify the specific source of this data: such as which part of the question, or which content of the previous step; and then determine the source and current use is consistent, the Now Step is used correctly.
Consistency Analysis: You need to check that the Now Step is consistent with the contents of the Previous Steps, and then you need to check that all the information inside the Now Step is consistent.
Calculation Analysis: If the Now Step involves any calculations, such as addition, subtraction, multiplication, division, equations, modulo operations, etc., you will first need to perform a check on the calculation, such as a reverse operation, to see if the calculation was done correctly, and then analyze the results of your check to see if there was an error in the calculation.
Conclusion:
Please verify the correctness of the Now Step based on your analysis, if there is any error in the Now Step then the Now Step is wrong and vice versa the Now Step is correct. At the end of the Conclusion, when you give your final answer, write it in the form "Verification: Is the step correct (Yes/No)? X", where X is either Yes or No.

Question: [Math Problem]
Previous Steps: [Previous Steps]
Now Step: [Current Step]
Please carefully analyze the correctness of the Now Step.
Reply:

Table 5: The Prompt to Construct Data