GroundedPRM: Tree-Guided and Fidelity-Aware Process Reward Modeling for Step-Level Reasoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Process Reward Models (PRMs) aim to improve multi-step reasoning in Large Language Models (LLMs) by supervising intermediate steps and identifying errors throughout the reasoning process. However, building effective PRMs remains challenging due to the lack of scalable, high-quality annotations. Existing approaches rely on costly human labeling, LLM-based self-evaluation prone to hallucination, or Monte Carlo (MC) estimation, which infers step quality solely from rollout outcomes, often introducing noisy and misaligned supervision due to credit misattribution. These issues result in three core limitations: noisy rewards, low factual fidelity, and misalignment with step-level reasoning objectives. To address these challenges, we introduce **GroundedPRM**, a tree-guided and fidelity-aware framework for automatic process supervision. To reduce reward noise and enable fine-grained credit assignment, we construct structured reasoning paths via Monte Carlo Tree Search (MCTS). To eliminate hallucinated supervision, we validate each intermediate step using an external tool, providing precise, execution-grounded correctness signals. To combine both step-level validation and global outcome assessment, we design a hybrid reward aggregation mechanism that fuses tool-based verification with MCTS-derived feedback. Finally, we format the reward signal into a rationale-enhanced, generative structure to promote interpretability and compatibility with instruction-tuned LLMs. GroundedPRM is trained on only 40K automatically labeled samples, amounting to just **10%** of the data used by the best-performing PRM trained with auto-labeled supervision. Nevertheless, it achieves up to a **26% relative improvement** in average performance on ProcessBench. When used for reward-guided greedy search, GroundedPRM outperforms even PRMs trained with human-labeled supervision, offering a scalable and verifiable path toward high-quality process-level reasoning.

1 Introduction

Large Language Models (LLMs) [1, 30, 9] have demonstrated impressive capabilities in planning [13, 42], decision-making [19], and complex task execution [36, 43]. However, they remain prone to hallucinations and reasoning errors, particularly in multi-step tasks such as mathematical problem solving. Existing methods like Chain-of-Thought prompting [35, 38] and Test-Time Scaling [26, 21] improve final accuracy, yet LLMs often produce solutions that appear coherent while containing errors in reasoning or calculation. These issues are further exacerbated by outcome-level supervision and coarse decoding strategies, e.g., majority voting, which overlook step-level correctness and provide little guidance during intermediate reasoning.

To mitigate these shortcomings, Process Reward Models (PRMs) have emerged as a promising direction [20]. PRMs assign step-level scores to reasoning trajectories, enabling fine-grained supervision

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

that supports better control and interpretability in multi-step reasoning. However, developing effective PRMs remains challenging due to the lack of reliable and faithful reward signals for training. Human annotation [20], while accurate, is costly and unscalable. LLM-as-a-judge [46] is more efficient but susceptible to hallucination, often rewarding fluent yet incorrect reasoning and thus compromising factual fidelity. Monte Carlo (MC) estimation [34, 22] provides another alternative by inferring step quality from final rollout outcomes, but it introduces noisy reward due to credit misattribution: correct steps may be penalized if the rollout fails, while flawed steps may be rewarded if the final answer happens to be correct [44]. Moreover, MC estimation typically evaluates only final outcomes, ignoring explicit assessment of intermediate step correctness, which misaligns the supervision signal with the objective of step-wise reasoning accuracy.

Several recent works have attempted to refine MC-based supervision, but core limitations persist. OmegaPRM [22] uses a binary search strategy to locate the first incorrect step, but still relies on rollout success to infer correctness, leaving credit assignment coarse. Qwen2.5-Math-PRM [44] filters samples based on agreement between MC estimation and LLM judgments, but this strategy inherits hallucination bias and scores each step solely based on rollout outcomes, without assessing whether it contributes to or hinders correct reasoning. BiRM [7] augments PRM with a value head to predict future success probability, but both reward and value signals are derived from noisy rollouts and lack external validation. These approaches offer partial improvements, yet remain constrained by outcome-based heuristics, hallucination-prone feedback, or weak step-level credit modeling.

To address these challenges, we propose **GroundedPRM**, a tree-guided and fidelity-aware framework for automatic process supervision. GroundedPRM is designed to resolve three core limitations in existing PRMs: noisy rewards, low factual fidelity, and misalignment with step-level reasoning objectives. First, to reduce reward noise and improve credit attribution, GroundedPRM leverages Monte Carlo Tree Search (MCTS) to construct structured reasoning paths and assess each step based on its contribution within the trajectory. Second, to ensure factual grounding, each intermediate step is verified using an external math tool, producing correctness signals based on executable logic rather than LLM-generated feedback, thereby eliminating hallucinated supervision. Third, to combine step-level validation with global outcome assessment, we design a hybrid reward aggregation mechanism that fuses tool-based verification with MCTS-derived feedback. Finally, all rewards are formatted into binary decisions paired with rationale-enhanced justifications, enabling interpretable supervision signals that are compatible with LLM-based generation and downstream reasoning workflows.

We evaluate GroundedPRM on ProcessBench and observe substantial gains in both data efficiency and overall performance. It is trained on only 40K automatically labeled samples, just **10%** of the data used by the best-performing PRM trained with auto-labeled supervision, yet achieves up to a **26% relative improvement** in average performance. Furthermore, when deployed in reward-guided greedy search, where candidate steps are selected based on predicted reward, GroundedPRM surpasses even PRMs trained with human-labeled supervision, establishing new state-of-the-art results across multiple mathematical reasoning benchmarks. These findings highlight the effectiveness, scalability, and practical value of our structured and fidelity-aware supervision framework for both training and inference.

The key contributions of this work are:

1. We propose GroundedPRM, a tree-guided and fidelity-aware process reward modeling framework that leverages MCTS to construct structured reasoning paths and support step-level credit assignment.
2. We introduce a fidelity-aware verification mechanism that validates each reasoning step using an external math tool, ensuring correctness grounded in executable logic and eliminating hallucinated supervision.
3. We design a hybrid reward aggregation mechanism that integrates tool-based step validation with feedback derived from MCTS-guided reasoning paths.
4. We format rewards into a rationale-enhanced, generative structure to improve interpretability and enable seamless integration into inference-time decoding and downstream reasoning workflows.
5. We demonstrate strong data efficiency and inference performance by evaluating GroundedPRM on ProcessBench and reward-guided greedy search.

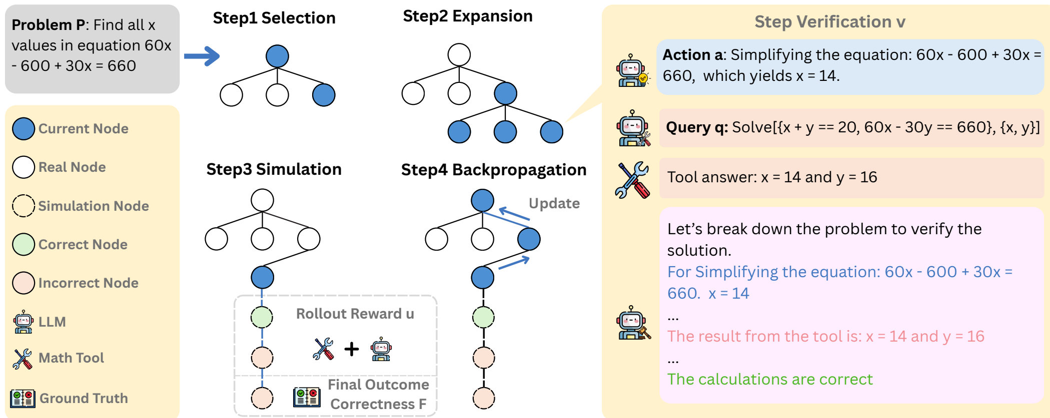

Figure 1: Overview of the GroundedPRM framework. GroundedPRM constructs reasoning paths via MCTS, where each node corresponds to an LLM-generated step. During simulation, intermediate steps are verified using symbolic tools, and final answers are checked against ground truth. Step-level and outcome-level correctness signals are aggregated into a rollout reward, which is backpropagated to guide search. The framework enables verifiable, interpretable, and structure-aware process supervision for multi-step reasoning. The generative rationale provides interpretable feedback for each step.

2 Related Work

2.1 Mathematical Reasoning with LLMs

Large Language Models (LLMs) have shown remarkable progress in solving math problems via Chain-of-Thought (CoT) reasoning, where step-by-step solutions often improve final answer accuracy [35]. Building on this, recent efforts have focused on enhancing reasoning capabilities through pretraining on math-related corpora [15, 24, 40], instruction tuning with annotated derivations [38, 19, 41, 39], and prompting strategies tailored for symbolic tasks [4, 17, 14]. Despite these improvements, LLMs remain vulnerable to intermediate reasoning errors, even when final answers are correct [45]. This discrepancy undermines the reliability of generated solutions, motivating the use of external verification or inference-time selection strategies [25, 31, 9]. Such approaches typically operate at the output level, offering limited supervision for correcting internal steps. Unlike prior methods that intervene at the output level, our approach supervises the reasoning process itself via step-level reward modeling, enabling finer-grained error identification and more faithful alignment with symbolic objectives.

2.2 Process Reward Models for Step-Level Supervision

To enhance reasoning fidelity and identify intermediate errors, PRMs have emerged as a promising alternative to outcome-level supervision [20, 32]. PRMs evaluate the correctness of individual reasoning steps and have been shown to improve alignment and generalization across math tasks [34, 44]. A key challenge lies in generating reliable step-level annotations. Early methods rely on expert-labeled datasets such as PRM800K [20], which are expensive to scale. Recent work has explored automatic synthesis through MC estimation [34, 22], often leveraging rollout outcomes to infer step validity. However, MC-based supervision introduces noise due to credit misattribution and dependency on the quality of the completion model [45, 44]. To mitigate this, several methods combine MC with LLM-as-a-judge consensus filtering [44] or adopt preference-based learning frameworks [6]. In contrast, our method GroundedPRM constructs PRM supervision from the ground up by integrating tree-structured search via MCTS [5], symbolic verification (via external math tools), and fused value-correctness reward modeling. This pipeline produces reward signals that are verifiable, structurally grounded, and directly aligned with symbolic reasoning objectives, addressing the core fidelity and alignment issues that prior methods leave unresolved.

3 Methodology

GroundedPRM is designed to address three core limitations of existing process reward modeling methods: noisy rewards, low factual fidelity caused by hallucinated self-assessment, and misalignment with step-level reasoning objectives. These challenges call for a framework that can assign fine-grained credit, validate the factual correctness of individual steps, and integrate local and global signals into a reliable and interpretable supervision objective. To this end, GroundedPRM introduces a tree-guided and fidelity-aware reward modeling framework composed of four core components. First, it employs Monte Carlo Tree Search (MCTS) to construct structured reasoning paths and assess each step based on its contribution within the search trajectory, enabling more stable and attribution-aware supervision than flat sampling-based methods. Second, it verifies each intermediate step using an external tool, producing binary correctness labels grounded in executable logic and thereby mitigating hallucinated feedback from the model. Third, it unifies verified step-level signals and final-answer correctness into a joint supervision objective, maintaining fine-grained credit assignment while offering stable and reasoning-grounded supervision. Finally, the reward supervision is formatted into a rationale-enhanced generative structure, pairing each step with both a binary score and an explanation to support interpretability and compatibility with instruction-tuned LLMs. An overview of this framework is illustrated in Fig. 1.

3.1 Tree-Guided Reasoning Path Construction

To enable stable and attribution-aware process supervision, GroundedPRM employs MCTS to construct structured reasoning paths for each input problem P. Each node in the search tree is associated with a partial reasoning state $s = \{s_1, \dots, s_i\}$, representing the sequence of previously generated reasoning steps. In addition to the state, each node stores auxiliary information including tool queries q, verification outcomes v, and value estimates Q. A reasoning step is represented as an action a, defined as a natural language expression generated by the LLM that extends the current reasoning state, transitioning it from state s to a new state s'. The value function $Q(s, a)$ estimates the expected return of applying action a in state s, and is updated through feedback from simulated rollouts. The search process consists of four stages:

Selection. Starting from the root node, the algorithm recursively selects child nodes according to a tree policy until reaching a node that is not fully expanded. To balance exploration and exploitation, we use the Upper Confidence Bound for Trees (UCT) [16], which balances estimated value with an exploration bonus that decreases as a node is visited more often, thereby encouraging the search toward promising yet under-explored nodes.

Expansion. If the selected node is not terminal, it is expanded by sampling up to m new actions from LLM, each producing a distinct child state s'. We set $m = 3$ in our experiments. This constrains the branching factor while maintaining reasoning diversity.

Simulation. From the newly expanded node, we simulate a complete reasoning trajectory by sequentially sampling steps $s_{i+1}, \dots, s_T$ until the model produces a final answer. We sample from the current state using the LLM in a left-to-right fashion to complete the solution. For each step s_j where $j \in \{i+1, ..., T-1\}$, we obtain a binary correctness label $v_j \in \{-1, 1\}$ using the tool-based verification procedure described in Section 3.2. Additionally, the final answer is compared against the ground-truth solution to determine the overall outcome $F \in \{-1, 1\}$. We adopt signed labels {-1,+1} instead of {0,1} so that incorrect steps propagate negative feedback, thereby decreasing node values during MCTS search rather than being treated as neutral. These per-step and final correctness signals are subsequently aggregated into a single rollout reward u, as defined in Section 3.3.

Backpropagation. The reward u computed for the simulated trajectory is propagated backward along the path traversed during selection. For each visited state-action pair (s_k, a_k) at depth d_k from the terminal node, we update its value as:

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \gamma^{d_k} \cdot (u_i + v_i), \tag{1}$$

where $k \in \{0, ..., i-1\}$, $\gamma \in (0, 1)$ is a decay factor controlling temporal discount, and d_k denotes the number of steps from the terminal node. This update scheme assigns stronger credit to steps closer to the final outcome, aligning attribution with their causal impact in the reasoning process.

By iteratively executing the four MCTS stages, GroundedPRM constructs a structured and diverse distribution over reasoning paths. This search process prioritizes trajectories with high step-level validity and globally correct outcomes, yielding supervision signals that are both structure-aware and attribution-sensitive. The resulting credit assignments are more stable and fine-grained than those produced by flat Monte Carlo rollouts, directly addressing reward noise and misattribution. Multiple rollouts are performed per input to balance path diversity with search efficiency.

3.2 Fidelity-Aware Step Verification with External Tool

To ensure reward fidelity and eliminate hallucinated supervision, GroundedPRM integrates symbolic verification into each reasoning step via external tools. During simulation (Section 3.1), the LLM generates a sequence of reasoning steps $\{s_{i+1}, \ldots, s_T\}$, where each s_j $(i+1 \leq j \leq T-1)$ denotes an intermediate reasoning step expressed in natural language during rollout.

For each step s_j, we construct a corresponding structured math query and submit it to a symbolic solver, such as Wolfram Alpha (WA). The tool's response is parsed to determine whether the computation or transformation expressed in s_j is factually correct. We represent this outcome as a binary verification label $v_j \in \{-1, 1\}$, where $v_j = 1$ indicates successful verification and $v_j = -1$ denotes failure. The resulting sequence $\{v_{i+1}, \ldots, v_{T-1}\}$ provides a fine-grained step-level correctness evaluation for the entire reasoning trace. These step-level signals are used during rollout to compute the aggregated reward u (Section 3.3). Unlike LLM-based self-evaluation, which often overestimates fluent but invalid reasoning, this fidelity-aware mechanism grounds supervision in objective, tool-based verification.

While WA is used in our experiments due to its strong symbolic reasoning capabilities, such as equation solving and equivalence checking, our verification module is tool-agnostic. It supports integration with alternatives like SymPy or domain-specific solvers. This modular design ensures that GroundedPRM generalizes across reasoning domains while maintaining high verification precision.

3.3 Hybrid Reward Aggregation

To construct reward signals that are both verifiable and forward-looking, GroundedPRM introduces a hybrid aggregation mechanism that combines symbolic step-level verification with trajectory-level outcome assessment. This design balances two supervision objectives: (1) factual fidelity of intermediate reasoning steps, and (2) global correctness of the final answer.

Given a simulated reasoning trace of length T, we collect step-level correctness signals $\{v_{i+1}, \ldots, v_{T-1}\}$, where each $v_i \in \{-1, 1\}$ is obtained via external tool verification (Section 3.2). In addition, we evaluate the final answer against ground truth to obtain a binary outcome signal $F \in \{-1, 1\}$. These signals are aggregated into a single scalar reward:

$$u_i = \frac{1}{T-1} \sum_{j=1}^{T-1} d_j \cdot v_j + \beta \cdot F, \tag{2}$$

where $\beta \geq 0$ is a weighting coefficient that adjusts the contribution of final answer correctness relative to step-level reliability. The resulting reward u is used during backpropagation in MCTS (Section 3.1) to update value estimates and guide exploration. We further define the MCTS value estimate at each state–action pair (s_i, a_i) as:

$$Q(s_i, a_i) = u_i + v_i. \tag{3}$$

By fusing local and global correctness signals, this hybrid reward formulation offers more stable and interpretable supervision than prior MC-based methods that rely solely on rollout success. Moreover, this mechanism directly addresses the three core limitations of existing PRMs: it reduces reward noise via structure-aware simulation, avoids unverifiable supervision through symbolic validation, and aligns the reward objective with both step-wise precision and task-level success.

3.4 Generative Process Reward Model

GroundedPRM adopts a generative reward modeling paradigm, enabling seamless integration with instruction-tuned LLMs and providing supervision for open-ended reasoning workflows. Each training instance is structured as a rationale-enhanced sequence that pairs intermediate reasoning steps with corresponding verification outcomes and justifications.

Formally, each instance includes: (1) the original problem P; (2) the full reasoning trajectory $\{s_1, \ldots, s_T\}$; (3) binary correctness labels $\{v_1, \ldots, v_T\}$ obtained via tool-based verification (Section 3.2); and (4) natural language explanations derived from external tool feedback.

Unlike conventional discriminative reward models that treat reward prediction as a binary classification task, we train GroundedPRM autoregressively to generate both correctness labels and rationales conditioned on the problem and its intermediate reasoning trace. This generative formulation improves interpretability and enables seamless integration into LLM-based reasoning pipelines.

3.5 Data Construction for GroundedPRM Training

To train GroundedPRM, we apply the full supervision framework described above to the MATH dataset [11], constructing a reward-labeled dataset with symbolic verification and hybrid scoring. For each problem, the policy model generates intermediate reasoning steps, which are verified using external tools (Section 3.2). Each step is labeled based on symbolic correctness, and the full trajectory is scored using the hybrid reward mechanism introduced in Section 3.3. To ensure coverage and diversity, we adopt a multi-round MCTS rollout strategy that explores both optimal and suboptimal paths. Post-processing includes filtering incomplete, inconsistent, or tool-unverifiable traces, and formatting the final data into a rationale-enhanced generative structure (Section 3.4). Each instance includes the problem, a full reasoning trace, correctness labels, and explanations. The resulting dataset contains approximately 40K verified samples, covering a broad spectrum of problem types and reasoning strategies with high symbolic fidelity.

4 Experiment

4.1 Experimental Setup

Benchmarks. We evaluate GroundedPRM from two perspectives: its ability to accurately identify erroneous steps within multi-step reasoning processes, and its effectiveness in directly enhancing downstream task performance.

- **ProcessBench [45].** This benchmark evaluates the ability of reward models to supervise step-level reasoning in mathematical problems. Each instance includes an LLM-generated solution with the first incorrect step annotated by human experts. Models are evaluated based on their ability to accurately identify the first faulty step or confirm that all steps are valid, following standard PRM evaluation protocols.
- **Reward-Guided Greedy Search.** To further assess the utility of GroundedPRM in guiding multi-step reasoning, we perform inference-time decoding using a reward-guided greedy strategy. At each generation step, we sample $N = 8$ candidate actions from Qwen2.5-7B-Instruct [23] using a temperature of 1, and select the candidate with the highest predicted reward assigned by the PRM. This process is repeated iteratively until a complete solution is generated. We evaluate this procedure on six mathematical benchmarks: AMC23 [3], AIME24 [2], MATH [11], College MATH [29], OlympiadBench [10], and Minerva MATH [18]. We also report the result of majority voting among eight samplings (maj@8), and pass@n, i.e., the proportion of test samples where any of the n samplings lead to the correct final answers.

Baselines. For both ProcessBench and reward-guided greedy search experiments, we compare GroundedPRM against the following representative baselines. These baselines span a diverse set of supervision strategies, including models trained with human-labeled rewards, automated annotations, and hybrid approaches, as well as a range of training data scales.

- **Math-Shepherd** [34]: Utilizes MC estimation to perform automated step-level annotation with hard labels.
- **RLHFlow-PRM series** [8]: Includes DeepSeek and Mistral variants, both of which use MC estimation for data generation, but adopt the Direct Preference Optimization (DPO) training paradigm.

Table 1: F1 scores on ProcessBench for models trained with auto-labeled data. Models marked with * share the same base model: Qwen2.5-Math-7B-Instruct. GroundedPRM achieves the highest average F1, surpassing the strongest existing model, Math-Shepherd-PRM-7B, by 26% relative improvement while using only 10% of the training data. All baseline results are directly cited from [44]. Oly. denotes OlympiadBench.

Model	#Sample	GSM8K	MATH	Oly.	Omni-MATH	Avg.
RLHFlow-DeepSeek-8B	253K	38.8	33.8	16.9	16.9	26.6
RLHFlow-Mistral-8B	273K	50.4	33.4	13.8	15.8	28.4
Qwen2.5-Math-7B-Math-Shepherd*	445K	**62.5**	31.6	13.7	7.7	28.9
EurusPRM-Stage1*	453K	44.3	35.6	21.7	23.1	31.2
EurusPRM-Stage2*	230K	47.3	35.7	21.2	20.9	31.3
Math-Shepherd-PRM-7B	445K	47.9	29.5	24.8	23.8	31.5
GroundedPRM	**40K**	43.4	**47.0**	**33.8**	**34.4**	**39.7**

- **Math-PSA-7B** [33]: Trained on mixed annotated data, namely PRM800K [20], Math-Shepherd [34], and generated data following [22].
- **EurusPRM-series** [27]: EurusPRM-Stage1 and EurusPRM-Stage2 constructs weakly supervised labels from final outcomes using noise-aware heuristics.
- **Qwen2.5-Math-7B series** [45, 44]: Qwen2.5-Math-7B-Math-Shepherd and Qwen2.5-Math-7B-PRM800K are trained with Math-Shepherd [34] and PRM800K [20] using Qwen2.5-Math-7B-Instruct [38], respectively.
- **Llemma-PRM800K-7B** [28]: Utilizes MC estimation to perform automated step-level annotation with hard labels.
- **ReasonEval-7B** [37]: Prompt-based model for evaluating step validity and redundancy.

Implementation Details. All reward models are fine-tuned on step-labeled reasoning trajectories using LoRA [12] for parameter-efficient adaptation. We use Qwen2.5-7B-Instruct [23] as the base model. The baseline methods adopt standardized prompt templates for critique generation, as detailed in Appendix, to ensure consistency in reward format and reasoning structure.

4.2 Results on ProcessBench

GroundedPRM Achieves Strong Supervision Performance with High Data Efficiency. As shown in Tab. 1 , GroundedPRM achieves the highest average F1 score among all PRMs trained with automatically labeled data, outperforming the second-best model, Math-Shepherd-PRM-7B, by a relative improvement of 26% while using only 10% training samples. GroundedPRM also ranks first on MATH, OlympiadBench, and Omni-MATH, indicating strong capability in evaluating complex symbolic reasoning steps. These results reinforce our central hypothesis: verifiable, structure-guided supervision is substantially more effective than scale alone. GroundedPRM's fidelity-aware rewards, grounded in symbolic tool validation and MCTS-based credit assignment, enable efficient learning under low-resource constraints.

Generative Supervision Enhances Interpretability and Robust Generalization. Unlike prior PRMs that produce only binary decisions, GroundedPRM adopts a generative format that outputs both a step-level reward and an accompanying rationale. This design improves alignment with instruction-tuned LLMs, encourages interpretable supervision, and enables the model to better distinguish between fluent but incorrect reasoning and truly valid logic. Empirically, GroundedPRM achieves notable improvements on challenging benchmarks like OlympiadBench and MATH, where fine-grained error localization is essential. These results suggest that explanation-based rewards foster more robust and generalizable reasoning behavior.

4.3 Analysis and Discussions

GroundedPRM Provides Superior Data Efficiency through Structured and Fidelity-Aware Supervision. To compare the effectiveness of our automatically labeled supervision against human-

Table 2: F1 scores on ProcessBench under different supervision strategies. GroundedPRM combines symbolic step validation with trajectory-level outcome assessment. Outcome-Only and Step-Only variants use single-source supervision, leading to misaligned or incomplete reward signals.

Scoring Strategy	GSM8K	MATH	OlympiadBench	OmniMATH	Avg.
Step-Only	40.1	42.3	28.3	29.2	35.0
Outcome-Only	1.4	3.3	1.0	1.0	1.7
GroundedPRM	**43.4**	**47.0**	**33.8**	**34.4**	**39.7**

Table 3: F1 scores of GroundedPRM and Qwen2.5-Math-7B-PRM800K under matched training sizes. Both methods are trained using Qwen2.5-7B-Instruct but differ in supervision sources. Despite relying solely on automatically labeled data, GroundedPRM consistently outperforms Qwen2.5-Math-7B-PRM800K across all data scales. Oly. denotes OlympiadBench.

#Sample	Model	GSM8K	MATH	Oly.	Omni-MATH	Avg.
10K	Qwen2.5-Math-7B-PRM800K	30.3	31.6	21.9	19.8	25.9
	GroundedPRM	**39.0**	**41.9**	**29.4**	**29.8**	**35.0**
20K	Qwen2.5-Math-7B-PRM800K	37.4	32.9	29.9	30.6	32.7
	GroundedPRM	**39.9**	**44.0**	**30.1**	**31.4**	**36.4**
30K	Qwen2.5-Math-7B-PRM800K	37.5	40.0	28.4	**34.8**	35.2
	GroundedPRM	**42.1**	**47.4**	**30.7**	31.7	**38.0**
40K	Qwen2.5-Math-7B-PRM800K	43.1	46.0	32.9	34.0	39.0
	GroundedPRM	**43.4**	**47.0**	**33.8**	**34.6**	**39.7**

labeled reward models under identical data budgets, we conduct a controlled comparison with the Qwen2.5-PRM series using the same model architecture, i.e., Qwen2.5-7B-Instruct, and matched training sizes. For each training size, we randomly sample a subset of examples to ensure a fair comparison. This setup isolates the effect of supervision quality by ensuring that both methods are evaluated under the same data scale. As shown in Tab. 3, GroundedPRM consistently achieves higher F1 scores across all training sizes, despite relying entirely on automatically constructed labels.

Dual-Signal Supervision Enhances Data Fidelity and Credit Attribution. To assess the contribution of our dual-signal supervision, we compare GroundedPRM against two ablations: Outcome-Only Supervision, which assigns labels based solely on final-answer correctness from MCTS rollouts, and Step-Only Supervision, which uses external tool verification without considering global trajectory outcomes. As shown in Tab. 2, Outcome-Only Supervision severely underperforms due to credit misattribution. Correct steps may be penalized if downstream steps fail, while flawed steps may be rewarded if the final answer happens to be correct. Step-Only Supervision achieves higher recall but suffers from precision loss, as symbolic tools can detect surface-level arithmetic errors but often fail to capture deeper logical flaws, resulting in false positives. In contrast, GroundedPRM fuses step-level correctness signals with trajectory-level feedback, enabling accurate credit assignment that is grounded in both local fidelity and global reasoning success. This hybrid design achieves the highest average F1, demonstrating the effectiveness of our supervision framework in producing reliable and structurally aligned reward signals.

4.4 Results on Reward-Guided Greedy Search

As shown in Tab. 4, GroundedPRM achieves the highest average accuracy across all PRMs under the reward-guided greedy search setting. Despite being trained on only 40K automatically labeled examples, it surpasses all PRMs trained on automated, mixed, or human-annotated data. GroundedPRM achieves new state-of-the-art results on AMC23 and matches or outperforms all baselines on MATH and Minerva MATH. These results confirm the effectiveness of GroundedPRM's design: symbolic verification improves fidelity, tree-based path construction ensures stable credit assignment, and rationale-enhanced generative supervision enables precise scoring under multi-candidate

Table 4: Accuracy of reward-guided greedy search using different PRMs to supervise the Qwen2.5-7B-Instruct policy model. GroundedPRM outperforms all PRMs trained with human, mixed, or automated labels, achieving the highest average accuracy. Oly. denotes OlympiadBench.

Model	**#Sample**	**AMC23**	**AIME24**	**MATH**	**College**	**Oly.**	**Minerva**	**Avg.**
pass@1	-	50.0	10.0	73.4	48.5	30.0	29.8	40.3
major@8	-	57.5	13.3	80.4	53.0	36.5	36.7	46.2
pass@8(Upper Bound)	-	82.5	20.0	90.4	61.0	48.0	49.6	58.6
Reward-Guided Greedy Search (prm@8)								
Trained on Human Annotated Data (PRM800K)								
Qwen2.5-Math-7B-PRM800K	264K	60.0	10.0	75.6	36.5	23.5	29.0	39.1
Llemma-PRM800K-7B	350K	42.5	6.7	72.2	47.5	27.6	29.5	37.7
ReasonEval-7B	350K	52.5	6.7	76.0	33.8	33.8	30.0	41.9
Trained on a Mix of Human and Automated Annotation Data								
Math-PSA-7B	860K	47.5	13.3	69.8	46.0	27.6	33.5	39.6
Trained on Automated Annotation Data								
Math-Shepherd-PRM-7B	445K	45.0	10.0	**74.8**	48.5	28.0	29.0	39.2
RLHFlow-Mistral-8B	253K	50.0	6.7	74.2	48.0	30.9	27.5	39.5
RLHFlow-Mistral-8B	273K	37.5	**13.3**	**74.8**	**50.5**	29.8	30.0	39.3
EurusPRM-Stage1	453K	47.5	10.0	73.0	49.0	30.1	31.0	40.1
EurusPRM-Stage2	230K	45.0	**13.3**	73.6	51.0	**31.6**	**32.5**	41.1
GroundedPRM	40K	**57.5**	10.0	**74.8**	49.0	31.3	**32.5**	**42.4**

decoding. By scoring each candidate step with accurate and verifiable feedback, GroundedPRM successfully guides the policy model toward accurate multi-step reasoning without requiring external demonstration or value-based lookahead.

5 Conlcusion

We introduced GroundedPRM, a tree-guided and fidelity-aware framework for process supervision. By combining structured path exploration via MCTS, symbolic step-level verification, hybrid reward aggregation, and rationale-enhanced supervision formatting, GroundedPRM addresses three core limitations of prior PRMs: low factual fidelity, noisy reward signals, and misalignment with step-level reasoning objectives. GroundedPRM is trained on only 40K automatically labeled samples, amounting to just 10% of the data used by the best-performing PRM trained with auto-labeled supervision. Nevertheless, it achieves up to a 26% relative improvement in average performance on ProcessBench. When used for reward-guided greedy search, GroundedPRM outperforms even PRMs trained with human-labeled supervision. These results underscore the effectiveness of structured, verifiable reward modeling in enhancing the reasoning capabilities of LLMs.

6 Future Work

While GroundedPRM provides a strong foundation for verifiable and structured process supervision, several directions remain open for further enhancement. Its performance can potentially benefit from stronger LLMs to improve trajectory quality. Expanding beyond a single symbolic tool could also extend the framework's applicability to more diverse reasoning domains. Additionally, integrating human preferences may further align supervision with interpretable and human-aligned reasoning. These extensions offer promising avenues to broaden the impact and generality of GroundedPRM in increasingly complex reasoning settings.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] AI-MO. Amc 2023, 2024b. *aimo-validation-aime*, 2024.

[3] AI-MO. Amc 2023, 2024b. *aimo-validation-amc*. https://huggingface.co/datasets/AI-MO/aimo-validation-amc, 2024. Accessed: 2025-07-30.

[4] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[6] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. *arXiv preprint arXiv:2406.10858*, 2024.

[7] Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Rui Zheng, Nijun Li, Tao Gui, Yun Li, et al. Better process supervision with bi-directional rewarding signals. *arXiv preprint arXiv:2503.04618*, 2025.

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

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[13] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey, 2024.

[14] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*, 2023.

[15] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[16] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[18] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

[19] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

[20] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[21] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

[22] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[23] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

[26] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[27] Lin Sun, Chuang Liu, Xiaofeng Ma, Tao Yang, Weijia Lu, and Ning Wu. Freeprm: Training process reward models without ground truth process labels. *arXiv preprint arXiv:2506.03570*, 2025.

[28] Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *Advances in Neural Information Processing Systems*, 37:51118–51168, 2024.

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

[30] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[31] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. *Hugging Face*, 2024.

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

[33] Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024.

[34] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

[35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[36] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

[37] Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27723–27730, 2025.

[38] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[39] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

[40] Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*, 2024.

[41] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[42] Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. Swarmagentic: Towards fully automated agentic system generation via swarm intelligence. *arXiv preprint arXiv:2506.15672*, 2025.

[43] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23378–23386, 2025.

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

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

[46] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the three limitations addressed (noisy rewards, low factual fidelity, misalignment) and the four components (MCTS, tool verification, hybrid aggregation, generative PRM), which match the methodology and results reported in Section 3 and Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This work is empirical/methodological without formal theorems; we provide algorithmic details and pseudocode.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Benchmarks, baselines, prompts, training hyperparameters, model family, and pseudocode are provided.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: We report datasets/benchmarks, sampling/search configs, LoRA settings, optimizer, LR schedule, seed, and hardware (4×A100 80GB) in Sec. 4 and App. B, App. C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report results averaged over 3 independent runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify GPU type and memory (4×A100 80GB) and main training configuration in App. B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public math datasets and does not involve personal data or human subjects; we follow dataset licenses and anonymity requirements.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss this in Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release a high-risk generative model or scraped sensitive data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all datasets/models.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release a new dataset. Supervision data are constructed from public sources with described pipeline.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing or human-subject studies are involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human-subject research is conducted.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are central to MCTS rollouts and generative PRM training; we specify the base model, prompts, and decoding settings in Section 4.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.