

CePRM: Generative Process Reward Modeling with Calibrated Values and Repair-Aware Supervision

Zongxuan Li* Yixiang Dong* Bo Hu Gonghao Cao Yanzhen Chen Huan Liu†

School of Computer Science and Technology, Xi'an Jiaotong University

ethan.li@stu.xjtu.edu.cn {yxdong, huanliu}@xjtu.edu.cn

Abstract

Generative process reward models (PRMs) improve step-level verification of mathematical reasoning, yet their effectiveness is limited by the quality of process supervision. The dominant Monte Carlo (MC) estimation derives step labels from final-answer accuracy rather than semantic correctness, producing systematic label noise, while standard pipelines discard the implicit tree structure of rollout data that encodes informative failure patterns. We propose CePRM, which addresses both limitations through Step-Cognitive MC Value Scaling, calibrating raw MC scores with step-level semantic signals via multiplicative adjustment, and pivot-step-directed repair mining, which exploits value cliffs in failed trajectories to construct same-prefix contrastive correction pairs via Sibling Advantage filtering. Experiments on ProcessBench and PRMBench show that CePRM, with a 4B backbone, achieves competitive or superior error localization and process verification against 7B-scale baselines, with notably strong error-side recognition.

1 Introduction

Large language models (LLMs) have made substantial progress in mathematical reasoning. Recent work has shifted from single-pass generation toward test-time scaling (TTS), where sampling multiple reasoning paths and evaluating intermediate steps, rather than only final answers, improves solution quality (OpenAI, 2024, 2025; Guo et al., 2025; Snell et al., 2025). Process reward models (PRMs) enable this paradigm by assigning step-level feedback to reasoning trajectories (Lightman et al., 2024; Snell et al., 2025). Generative PRMs further produce explanatory rationales before each judgment (Zhang et al., 2026; Zhao et al., 2026), supporting interpretable verification, but their effectiveness hinges on the quality of process supervision used for training.

The dominant supervision strategy derives step-level labels from Monte Carlo (MC) estimation: from each reasoning prefix, multiple continuations are sampled, and final-answer accuracy serves as the step value (Wang et al., 2024). This value measures the probability that a prefix eventually yields the correct answer (Ding et al., 2025), not whether the current step is itself semantically correct. The mismatch creates two systematic biases: incorrect steps are overestimated when later rollouts happen to recover the answer, and correct steps are underestimated when subsequent continuations fail (Xie et al., 2026). Such label noise propagates directly into the verification rationales synthesized from these scores.

A complementary limitation is that flattening rollout data into independent sequences preserves within-trajectory order but obscures which trajectories share a parent and where their branches diverge (Yao et al., 2023; Zhang et al., 2024). CePRM retains this topology to select candidate pivots and sample replacement siblings from a shared prefix (Section 3.2). In some failed solutions, the step value drops sharply along the trajectory; we call this pattern a *value cliff* (Luo et al., 2024; Yang et al., 2025b). Empirically, value cliffs appear in 32.6% of problems with a mean drop of 0.86 and are concentrated in the first few reasoning steps (Figure 1). We use the earliest qualifying discontinuity as a repair candidate; we do not evaluate its agreement with human first-error annotations.

Motivated by these observations, we propose **CePRM**, a generative PRM whose training supervision addresses both MC label noise and underexploited rollout topology through two complementary mechanisms. *Step-Cognitive MC Value Scaling* calibrates raw MC scores by incorporating step-level semantic signals through a multiplicative adjustment anchored to the original estimate, reducing noise where rollout variance is highest. *Pivot-step-directed repair mining* locates the first value

* Equal contribution. † Corresponding author.

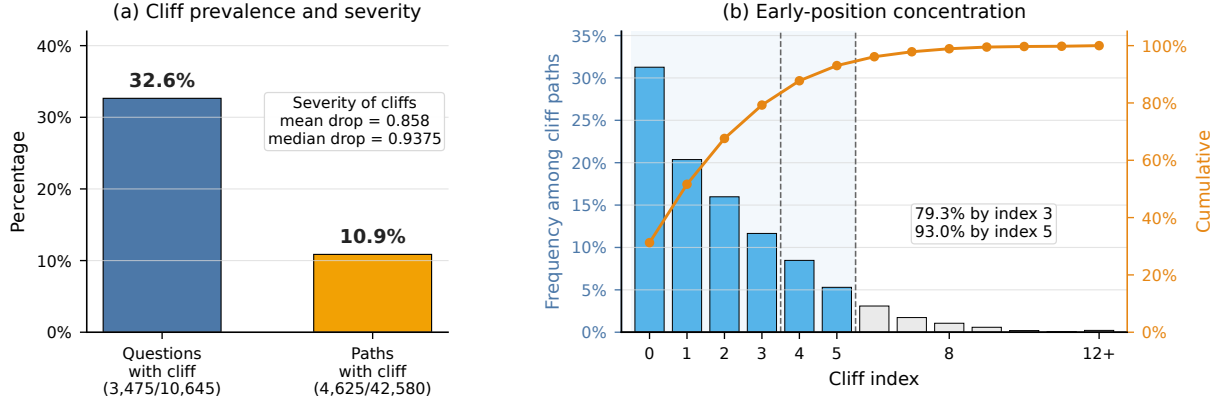


Figure 1: Value cliffs in MC rollout trees. (a) In our rollout pool, 32.6% of questions (3,475/10,645) and 10.9% of paths (4,625/42,580) exhibit a value cliff, with a mean (median) severity of 0.858 (0.938). (b) Cliff positions concentrate early: 79.3% occur by step index 3 and 93.0% by step index 5, making the first few reasoning steps the primary target for pivot-step-directed repair mining.

cliff in each failed trajectory, then samples and verifies corrective continuations under the same reliable prefix. Together, these mechanisms produce structured supervision spanning correct, erroneous, and repair-aware verification instances.

Our contributions are as follows:

- We present CePRM, a generative PRM trained from MC-anchored, semantically informed process values and repair-aware supervision.
- CePRM selects the earliest qualifying value discontinuity as a candidate pivot, samples replacement siblings from its parent prefix, and retains repairs that satisfy absolute-value and Sibling Advantage constraints.
- Within the evaluated Qwen-based pipeline, CePRM-4B achieves first-error localization and fine-grained process verification competitive with representative 7B-scale PRM baselines on ProcessBench and PRMBench, with particularly strong error-side recognition.

2 Related Work

Discriminative Process Reward Models. PRMs assign step-level feedback to reasoning trajectories, supporting reranking, search, and error localization (Lightman et al., 2024; Snell et al., 2025; Ma et al., 2023; Zhang et al., 2024; Zheng et al., 2025). Early discriminative PRMs predict a scalar score or binary label per step (Uesato et al., 2022). PRM800K (Lightman et al., 2024) demonstrates the value of high-quality human-annotated process supervision, but step-wise annotation is difficult to

scale. Later methods therefore derive automatic labels from Monte Carlo rollouts (Wang et al., 2024; Luo et al., 2024), and improve label construction with first-error localization, tree search, or consensus filtering (He et al., 2024; Luo et al., 2024; Zhang et al., 2025). SCAN (Ding et al., 2025) mitigates MC noise through learned self-denoising, without leveraging step-level semantic signals or same-prefix repair contrasts. Despite their effectiveness, these methods still depend on final-answer correctness, making reward value sensitive to rollout bias and label noise.

From Discriminative to Generative PRMs. Because discriminative PRMs yield only scalar scores without explanatory judgments, recent work introduces generative PRMs that output verification rationales before a correctness decision (Xiong et al., 2024; Wei et al., 2022). GenPRM (Zhao et al., 2026) casts process supervision as a generative verification task with code-execution feedback. Related approaches explore reasoning-driven, multimodal, and general-purpose generative reward modeling (She et al., 2025; Zhang et al., 2026; Wang et al., 2026), while rStar-Math develops self-evolved process preference modeling (Guan et al., 2025). CePRM instead focuses on supervision construction: it adjusts MC values with step-level semantic signals and mines same-prefix repair pairs from rollout trees. First-error annotation provides an error position, while PRM-guided tree search collects or ranks trajectories. CePRM retains an outcome-verified failed/repaired path pair with a shared parent prefix and converts the pair into generative supervision.

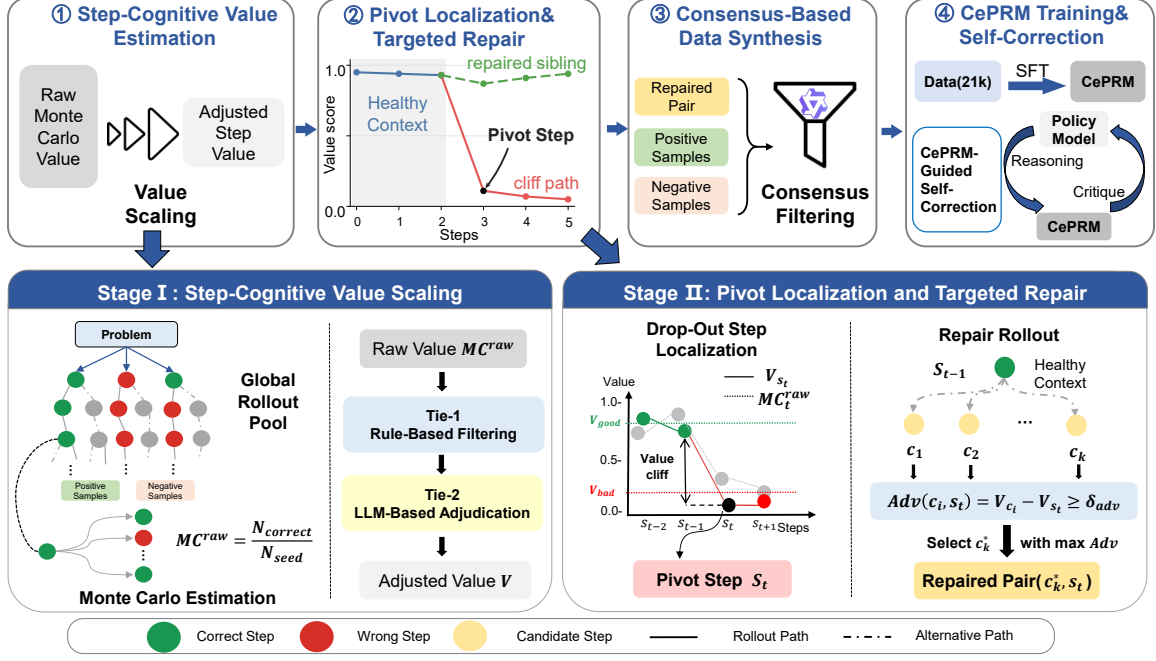


Figure 2: Overview of the CePRM pipeline. Step-Cognitive Value Scaling adjusts raw MC scores using rule checks and LLM-based adjudication. Pivot localization and targeted repair identify value cliffs in failed trajectories and construct same-prefix correction pairs through Sibling Advantage filtering. Please refer to Section 3 for details.

3 Methodology

3.1 Step-Cognitive MC Value Scaling

Process Reward Models (PRMs) rely on step-level supervision to evaluate intermediate reasoning. In the absence of large-scale human annotations, Monte Carlo (MC) estimation is the dominant strategy for constructing step-level process labels (Wang et al., 2024). Given a problem x , ground-truth answer y^* , and reasoning trajectory $\gamma = (a_1, \dots, a_T)$, we denote the prefix state after step t as $s_t = [x, a_1, \dots, a_t]$ and compute the raw MC score by sampling K continuations:

$$MC_t^{raw} = \frac{1}{K} \sum_{k=1}^K \mathbb{I}(\text{Ans}(\rho_t^{(k)}) = y^*), \quad (1)$$

where $\rho_t^{(k)}$ is the k -th complete solution sampled from s_t and $\text{Ans}(\cdot)$ extracts the final answer.

Because MC_t^{raw} measures the probability that a prefix eventually yields the correct answer rather than whether the current step is itself correct (Ding et al., 2025), it systematically overestimates incorrect steps that are later repaired and underestimates correct steps followed by failed continuations. We define *local mathematical correctness* as the validity of the current inference or transformation given the problem and the preceding valid prefix.

Relevance, variable consistency, coherence, and expression well-formedness are auxiliary signals, not separate definitions of correctness.

Step-Cognitive MC Value Scaling adjusts MC_t^{raw} using step-level semantic signals. The resulting value V_t is used for pivot localization, sample selection, and filtering; it is not interpreted as a calibrated probability or a ground-truth correctness label.

Semantic Assessment: Value adjustment has two components. Four rules check every step for problem relevance, variable-reference consistency, safely evaluable numeric/equation validity, and explicit expression anomalies. Each rule independently returns +0.1 for directly verified positive evidence, −0.1 for an explicit violation, or 0 for abstention/not-applicable; multiple nonzero outputs may co-occur and are summed, while merely failing to detect a violation is neutral. QwQ-32B is invoked only for steps in the ambiguous raw-MC range, steps flagged by a rule, and steps with a large adjacent raw-score drop. For each selected step, QwQ assigns a signed weight for at most one dominant cognitive dimension and returns 0 when local evidence is insufficient; the five dimensions are not accumulated. For unselected steps, $\omega_t^{llm} = 0$. Details are provided in Appendix B.2.

Cognitive Scaling Function: Given these sig-

nals, the calibrated step value is

$$V_t = \text{Clip} \left(MC_t^{\text{raw}} \cdot \left(1 + \lambda \left[\sum_j \omega_{t,j}^{\text{rule}} + \omega_t^{\text{llm}} \right] \right), 0, 1 \right), \quad (2)$$

where λ controls calibration strength and $\text{Clip}(\cdot, 0, 1)$ constrains the output to $[0, 1]$.

The multiplicative form ensures two properties absent from an additive alternative ($V = MC^{\text{raw}} + \delta$): *zero-preservation*, so that unanimous rollout failure cannot be overridden by semantic signals, and *proportional scaling*, so that adjustment magnitude is governed by MC_t^{raw} itself rather than applied uniformly. V_t thus serves as an MC-anchored process value for downstream pivot-step localization, repair mining, and supervision filtering, not as a standalone semantic correctness label.

3.2 Pivot-Step-Directed Repair Mining

Calibrated step values indicate where reasoning fails but not how to correct it. Richer supervision can be obtained by pairing an erroneous step with an outcome-verified correction under the same reliable prefix, producing a same-context contrastive signal. CePRM extracts such pairs through *pivot-step-directed repair mining*: it locates the first step that triggers a value cliff and mines corrective siblings from the shared parent context.

Rollout Tree and Value Cliff: Multiple rollouts from the same problem share common prefixes and naturally form a tree $\mathcal{T}_x$. Many failed trajectories in this tree are not erroneous from the outset: they maintain high calibrated value until a single step, caused by a logical shift, condition misuse, or conceptual error, triggers a sharp value collapse. We term this collapse a *value cliff* and the first step that triggers it the *pivot step*. In the rollout pool, value cliffs appear in 32.6% of problems and are concentrated in early reasoning steps (Figure 1). We do not evaluate their agreement with human first-error annotations.

Pivot-Step Localization: We now formalize cliff detection. Let $\mathcal{P}_x$ denote the rollout pool for problem x . CePRM searches for pivot steps only in trajectories with incorrect final answers. For a failed trajectory $\gamma = (a_1, \dots, a_T)$, we define the adjacent value drop $\Delta_t = V_{t-1} - V_t$ and select the pivot step by

$$t^* = \min \left\{ t \mid V_{t-1} \geq \tau_{\text{high}}, (V_t \leq \tau_{\text{low}} \vee \Delta_t \geq \delta_{\text{drop}}) \right\}. \quad (3)$$

The threshold τ_{high} ensures a reliable parent prefix; the disjunction captures both absolute value collapse ($V_t \leq \tau_{\text{low}}$) and sharp relative drops ($\Delta_t \geq \delta_{\text{drop}}$) that may not reach τ_{low} but still indicate a critical error. The minimization selects the earliest step satisfying these conditions. We treat it as a candidate pivot, not a verified first-error annotation. Trajectories with no qualifying step are discarded for lacking a reliable prefix.

Sibling Repair and Verification: After locating t^* , CePRM fixes the reliable parent prefix $c = s_{t^*-1}$ and samples candidate replacement steps from the rollout model p_θ :

$$\hat{a}_{t^*}^{(m)} \sim p_\theta(\cdot \mid c), \quad m = 1, \dots, N_{\text{repair}}. \quad (4)$$

Each candidate forms a sibling prefix $\hat{s}_{t^*}^{(m)} = [c, \hat{a}_{t^*}^{(m)}]$ of the prefix $s_{t^*} = [c, a_{t^*}]$. Every sibling prefix is independently extended via rollout and scored with calibrated values (Eq. 2); budgets are detailed in Appendix B.3.

Sibling Advantage: Absolute candidate value can be confounded by problem difficulty or prefix quality. Following advantage-based reasoning (Zhao et al., 2026; Shao et al., 2024), we compare siblings under the same parent:

$$\text{Adv}(\hat{s}_{t^*}^{(m)}, s_{t^*}) = V(\hat{s}_{t^*}^{(m)}) - V(s_{t^*}). \quad (5)$$

This difference isolates the local value gain from replacing the pivot step. We use subtraction rather than division to avoid instability when $V(s_{t^*}) \approx 0$. A candidate is accepted only if it satisfies both a stability and an advantage constraint:

$$V(\hat{s}_{t^*}^{(m)}) \geq \tau_{\text{pos}}, \quad \text{Adv}(\hat{s}_{t^*}^{(m)}, s_{t^*}) \geq \delta_{\text{adv}}. \quad (6)$$

The first condition ensures absolute reliability of the repaired prefix; the second requires a significant improvement over the erroneous step. Among accepted candidates, CePRM selects the one with the largest advantage:

$$m^* = \arg \max_m \text{Adv}(\hat{s}_{t^*}^{(m)}, s_{t^*}). \quad (7)$$

Path-level definitions of the resulting correction pairs appear in Appendix B.5. The dual filtering ensures each accepted pair provides not only a correct final answer but also a verified local value gain over the erroneous step, yielding same-prefix contrastive supervision for generative PRM training.

3.3 Consensus Synthesis and Training

The preceding components yield calibrated step values (Section 3.1) and same-prefix contrastive repair pairs (Section 3.2); converting these heterogeneous signals into training supervision requires structured rationale synthesis and quality filtering. For a trajectory $\gamma = (a_1, \dots, a_T)$, we identify the first step whose calibrated value falls below a negative threshold:

$$e(\gamma) = \min\{t \mid V_t \leq \tau_{\text{neg}}\}, \quad (8)$$

setting $e(\gamma) = -1$ when no such step exists. Let $\mathbb{G}$ denote the set of accepted path-level correction pairs, each comprising a shared reliable prefix, the original failed branch, an outcome-verified repaired continuation, and the pivot index t^* (formal definition in Appendix B.5).

Rationale Synthesis: Prior work improves synthesized reasoning supervision through program-based error checking, while generative PRMs use explicit rationales and code verification for step-level judgments (Zhu et al., 2024; Zhao et al., 2026). We use the same QwQ-32B model (Qwen Team, 2025) to generate a structured rationale $z_t = \langle \text{analyze}, \text{fix}, \text{output} \rangle$ for each step a_t , conditioned on the problem x and prefix $(a_1, \dots, a_{t-1})$. The analyze field provides step-level verification reasoning; fix supplies a revision suggestion directly leveraging the contrastive repair pairs from Section 3.2; output predicts a binary correctness judgment.

Consensus Filtering: Teacher-generated rationales may be inconsistent with process labels (Zheng et al., 2023; Zhao et al., 2026). Consensus filtering uses deterministic parsing, not another QwQ call, to discard samples whose generated fields disagree with the process label or repair target. After generating all step-level rationales for a trajectory, the teacher outputs its predicted first-error position $\tilde{e}$ (or $\tilde{e} = -1$ if the trajectory is judged fully correct). A sample is retained only when this prediction agrees with the process labels: $\tilde{e} = -1$ for positive samples; $\tilde{e} = e(\gamma)$ for negative samples; and for repair-aware samples from $\mathbb{G}$, $\tilde{e} = t^*$, with the additional requirement that the suggested fix reaches the correct final answer from the shared prefix.

Training: The filtered supervision comprises three partitions:

$$\mathcal{D} = \mathcal{D}_{\text{pos}} \cup \mathcal{D}_{\text{neg}} \cup \mathcal{D}_{\text{rep}}. \quad (9)$$

CePRM is fine-tuned with standard autoregressive cross-entropy to generate z_t and a step-level judgment given the prefix state $s_t = [x, a_1, \dots, a_t]$.

4 Experiment

4.1 Experimental Setup

Benchmarks and baselines: We evaluate on ProcessBench (Zheng et al., 2025), which tests first-error localization in step-by-step solutions, and PRMBench (Song et al., 2025), which measures fine-grained process verification across multiple dimensions; further details are provided in Appendix A. For ProcessBench, *err.* and *corr.* are accuracies on erroneous and fully correct trajectories, and F1 is their harmonic mean. Baselines span both discriminative PRMs (*Math-Shepherd* (Wang et al., 2024), *RLHFlow* (Xiong et al., 2024), *Skywork-PRM* (He et al., 2024), *ReasonEval-7B* (Xia et al., 2025), *Qwen2.5-Math-PRM* (Zheng et al., 2025; Zhang et al., 2025)) and generative PRMs (*R-PRM* (She et al., 2025), *GenPRM* (Zhao et al., 2026)). Entries marked with * in Tables 1–2 are published values; the unmarked GenPRM-7B and CePRM-4B entries are author-evaluated. These cross-source comparisons do not imply identical evaluation budgets.

Data construction: Rollouts are sampled from Qwen2.5-Math-7B-Instruct (Yang et al., 2024) on the MATH training split (Hendrycks et al., 2021) with difficulty-adaptive sampling (Appendix B.1), retaining at most four paths per problem and drawing 25 conditional continuations per prefix to compute MC^{raw} . Before synthesis, the pool contains 22,812 positive and 3,690 negative trajectories plus 3,541 path-level repair pairs; these heterogeneous source units are not final SFT instances. QwQ-32B (Qwen Team, 2025) performs adjudication and rationale synthesis; deterministic parsing and consensus filtering yield 21,421 training and 1,000 validation SFT conversations. Construction hyperparameters are listed in Appendix B, and the prompts are provided in Appendix E.

Training: CePRM is fine-tuned from Qwen3-4B (Yang et al., 2025a) to generate structured `<analyze>`, `<fix>`, and `<output>` fields for each reasoning step; full optimization settings are provided in Appendix B.6. Training data are drawn exclusively from the MATH training split and do not overlap with the ProcessBench/PRMBench evaluation sets.

Model	GSM8K			MATH			OLYMPIADBENCH			OMNIMATH			Avg. F1
	err.	corr.	F1	err.	corr.	F1	err.	corr.	F1	err.	corr.	F1	
<i>LLM-as-judge, proprietary models</i>													
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
<i>LLM-as-judge, open-source models</i>													
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
<i>PRMs</i>													
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
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
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*	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
GenPRM-7B	67.7	94.0	78.7	74.6	87.0	80.3	68.3	76.6	72.2	63.5	77.4	69.8	75.2
CePRM-4B	75.0	88.8	81.3	76.0	78.4	77.2	71.2	67.1	69.1	72.0	63.1	67.2	73.7

Table 1: Performance on ProcessBench. *err.* and *corr.* denote accuracy on erroneous and fully correct trajectories, respectively; F1 is their harmonic mean. Results marked with * are published values from [She et al. 2025](#); the unmarked GenPRM-7B and CePRM-4B values are author-evaluated. Bold marks the best PRM result.

Model	Simplicity			Soundness					Sensitivity				Overall
	NR.	NCL.	Avg.	ES.	SC.	DC.	CI.	Avg.	PS.	DR.	MS.	Avg.	
<i>LLM-as-judge, proprietary models</i>													
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
<i>PRMs</i>													
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
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
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*	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
GenPRM-7B	50.8	58.7	54.8	64.7	71.5	70.3	67.4	68.5	60.7	66.6	99.3	75.4	65.9
CePRM-4B	50.2	60.1	55.2	69.0	71.9	70.5	74.6	71.5	59.5	65.6	99.5	74.9	68.2

Table 2: Performance on PRMBench. Metric abbreviations are defined in Appendix A. Results marked with * are published values from [She et al. 2025](#); the unmarked GenPRM-7B and CePRM-4B values are author-evaluated. Bold marks the best PRM result.

4.2 Main Results

CePRM achieves strong error localization with a smaller backbone. Table 1 shows that CePRM-4B achieves an average F1 of 73.7 on ProcessBench, matching Qwen2.5-Math-PRM-7B (73.5) and surpassing R-PRM-7B (70.4) with a 4B backbone against 7B-scale baselines, while approaching GenPRM-7B (75.2). CePRM-4B has the highest error-side accuracy among the PRMs on all four

subsets: 75.0 on GSM8K, 76.0 on MATH, 71.2 on OlympiadBench, and 72.0 on OmniMath. This strength comes at the cost of lower correct-side scores, reflecting more selective step acceptance consistent with the effect of repair-oriented supervision (Section 4.4).

CePRM yields balanced gains across verification dimensions. On PRMBench (Table 2), CePRM-4B obtains the highest overall score among PRMs

(68.2), surpassing R-PRM-7B (66.8), GenPRM-7B (65.9), and Qwen2.5-Math-PRM-7B (65.5). The gain is most pronounced in soundness, where CePRM-4B achieves the best SC and DC scores (71.9, 70.5), and in simplicity, with the highest NCL score (60.1). Its sensitivity is slightly below R-PRM-7B, consistent with conservative calibration of Step-Cognitive MC Value Scaling, which refines MC-derived estimates rather than overriding them with unconstrained semantic signals. On a held-out Best-of-8 evaluation, CePRM improves macro accuracy over majority voting from 35.59 to 37.72 (Appendix C).

4.3 Effect of Value Scaling on Label Quality

Beyond downstream benchmark gains, we verify that value scaling directly improves supervision quality. We measure label noise as the disagreement rate between threshold-discretized MC labels and reference step-level annotations on the training data. Figure 3 shows that scaling reduces total label noise by 11.0% relative to raw MC labels, with a larger reduction for false-positive noise (invalid steps labeled correct; 11.4%) than for false-negative noise (valid steps labeled incorrect; 4.0%). Because Eq. 2 preserves $MC_t^{raw} = 0$, semantic calibration cannot promote zero-valued prefixes; excluded outcome-correct paths are withheld from positive supervision, not mislabeled (Appendix D).

The improvement is robust to threshold choice. Figure 4 sweeps $\tau_{neg} \in [0, 0.4]$ with $\tau_{pos} = 1 - \tau_{neg}$. Noise increases for both raw and scaled labels as τ_{neg} grows, but the scaled curve remains below the raw curve at every tested threshold. Scaling further reduces near-threshold cases by 24.0%, indicating that calibration shifts ambiguous prefixes away from the decision boundary, a property expected to benefit downstream pivot localization and Sibling Advantage filtering.

Par.	Components			F1				
	CoT	VS	DR	GSM8K	MATH	OLY.	OMNI.	Avg.
Disc.	×	×	×	59.4	60.5	53.8	53.5	56.8
Disc.	×	✓	×	69.3	65.7	53.6	52.1	60.2
Gen.	✓	×	×	72.8	71.1	61.4	57.0	65.6
Gen.	✓	×	✓	76.1	72.0	63.5	62.0	68.4
Gen.	✓	✓	✓	81.3	77.2	69.1	67.2	73.7

Table 3: Component-wise ablation on ProcessBench. Par.: verification paradigm (Disc. = discriminative, Gen. = generative); VS: Step-Cognitive MC Value Scaling; DR: pivot-step-directed repair mining.

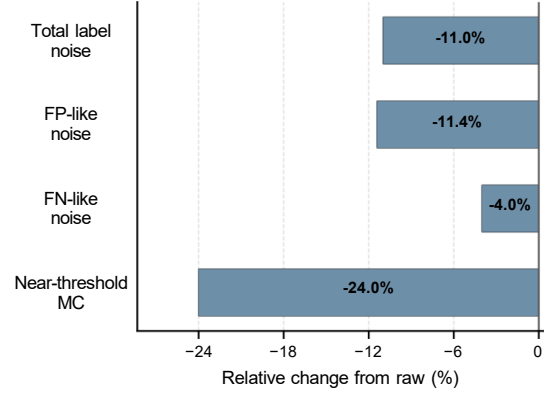


Figure 3: Effect of Step-Cognitive MC Value Scaling on MC label noise.

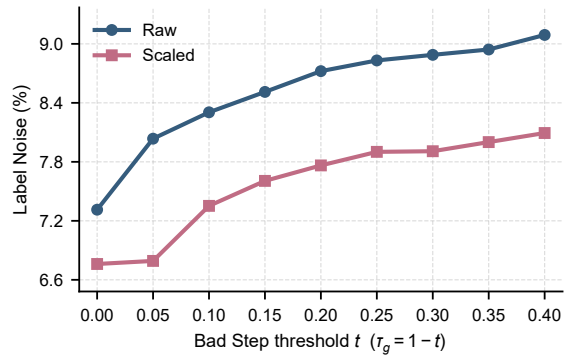


Figure 4: Sensitivity of label noise to the bad-step threshold t , with $\tau_g = 1 - t$.

4.4 Ablation Study

Table 3 compares matched CePRM configurations using generative verification rationales (CoT), Step-Cognitive MC Value Scaling (VS), and pivot-step-directed repair mining (DR).

Generative rationales establish a strong baseline.

The generative CoT configuration obtains an average F1 of 65.6, compared with 56.8 for the discriminative baseline (+8.8). We treat this generative CoT setting as the reference baseline for all subsequent comparisons. The generative configurations use the same QwQ-32B teacher, Qwen3-4B backbone, and SFT optimization; only their supervision composition differs.

Value scaling yields limited gains in isolation.

Under the discriminative paradigm, VS improves average F1 by 3.4 points, with clear gains on GSM8K (+9.9) and MATH (+5.2) but slight decreases on OlympiadBench and OmniMath. This asymmetry is consistent with the multiplicative form of Eq. 2: calibration can suppress MC noise

Problem: The perimeter of equilateral $\triangle PQR$ is 12. The perimeter of regular hexagon $STUVWX$ is also 12. What is the ratio of the area of $\triangle PQR$ to the area of $STUVWX$?

Path:...

Step 7: ... We already calculated the area of one equilateral triangle in the previous step; now we multiply that by 6: $\text{Area}_{\text{hexagon}} = 6 \times 4\sqrt{3} = 24\sqrt{3}$

Qwen2.5-Math-PRM-7B: prediction = -1, match = False, judgement = Yes

GenPRM-7B: prediction = 5, match = False, judgement = No

CePRM: prediction = 6, match = True, judgement = No

<analyze> Incorrectly states that the regular hexagon’s area is 6 times the area of one equilateral triangle. The regular hexagon’s area is not 6 times the area of an equilateral triangle with side length 2... <analyze> <fix>The calculation is wrong. Correct step: To find the area of the regular hexagon, we use the formula for the area of a regular hexagon with side length t : $\text{Area}_{\text{hexagon}} = (\frac{3\sqrt{3}}{2}) \times t^2$. Substituting $t = 2$: $\text{Area}_{\text{hexagon}} = (\frac{3\sqrt{3}}{2}) \times 2^2 = (\frac{3\sqrt{3}}{2}) \times 4 = 6\sqrt{3}$.<fix> <output>: No <output>

Figure 5: Case study of CePRM as a generative critic. The model identifies the first erroneous step, explains the reasoning error, and provides an actionable correction.

where rollout estimates are relatively stable, but scalar value adjustment alone does not address the harder benchmarks’ core supervision bottleneck, locating and repairing local reasoning failures.

Value scaling and repair mining are complementary. The CoT+DR configuration obtains an average F1 of 68.4, with its largest difference from CoT on OmniMath (+5.0). Combining all three components yields an average F1 of 73.7, an 8.1-point gain over the CoT baseline, with consistent improvements across all four subsets and particularly large margins on OlympiadBench (+7.7) and OmniMath (+10.2). The difference associated with VS is +5.3 between CoT+DR and CoT+VS+DR, compared with +3.4 between the two discriminative configurations. These configuration comparisons do not isolate teacher strength, data filtering, or model-family effects.

4.5 Case Study

As shown in Figure 5, the path incorrectly reused the area of $\triangle PQR$ when computing the area of the regular hexagon, leading to an erroneous result of $24\sqrt{3}$. Qwen2.5-Math-PRM-7B failed to detect this mistake and judged the step as correct, while GenPRM-7B detected an error but failed to locate the correct pivot step. In contrast, CePRM accurately identified the erroneous step and further explained that the hexagon should be computed using side length 2, yielding the correct area $6\sqrt{3}$. This case demonstrates that CePRM can not only detect incorrect reasoning steps, but also provide

faithful error localization and correction-oriented analysis.

5 Conclusion

We propose **CePRM**, a generative process reward model trained on step-cognitive and correction-oriented process supervision. To address noisy reward labels in conventional Monte Carlo estimation and the lack of repair-oriented supervision for erroneous steps, CePRM introduces step-cognitive semantic calibration to refine raw MC scores under step-level semantic constraints. It further exploits value cliffs in the rollout tree to locate critical reasoning failures, and constructs locally contrastive correction supervision through targeted repair and Sibling Advantage filtering. The resulting generative PRM can not only identify erroneous reasoning steps, but also provide revision-oriented analysis and feedback. Experiments demonstrate the effectiveness of CePRM in process-level error localization and fine-grained process verification. These results suggest that combining semantic calibration, rollout-level topology, and correction-oriented supervision is a promising direction for improving process supervision in mathematical reasoning.

Limitations

Computational constraints prevented controlled teacher-size ablations and broader cross-family evaluation. Our cognitive adjudication and rationale synthesis use a fixed QwQ-32B teacher within a Qwen-based pipeline, so the current results can-

not fully disentangle the supervision design from teacher strength. In addition, value cliffs are concentrated in early reasoning steps, while localization degrades for later errors. CePRM may therefore be less reliable on long trajectories or late-stage failures.

References

- Yuyang Ding, Xinyu Shi, Juntao Li, Xiaobo Liang, Zhaopeng Tu, and Min Zhang. 2025. [SCAN: Self-denoising monte carlo annotation for robust process reward learning](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 54005–54033. Curran Associates, Inc.
- Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. [rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 20640–20661. PMLR.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning](#). *Nature*, 645:633–638.
- Jujie He, Tianwen Wei, Rui Yan, Jiakai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024. [Skywork-o1 Open Series](#).
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.
- 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. 2024. [Improve mathematical reasoning in language models by automated process supervision](#). *arXiv preprint arXiv:2406.06592*.
- Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. [Let’s reward step by step: Step-level reward model as the navigators for reasoning](#). *arXiv preprint arXiv:2310.10080*.
- OpenAI. 2024. Learning to reason with LLMs. <https://openai.com/index/learning-to-reason-with-llms/>.
- OpenAI. 2025. OpenAI o3-mini. <https://openai.com/index/openai-o3-mini/>.
- Qwen Team. 2025. QwQ-32B: Embracing the power of reinforcement learning. <https://qwenlm.github.io/blog/qwq-32b/>.
- 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](#). *arXiv preprint arXiv:2402.03300*.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. [R-PRM: Reasoning-driven process reward modeling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451, Suzhou, China. Association for Computational Linguistics.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. [Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning](#). In *International Conference on Learning Representations*, volume 2025, pages 10131–10165.
- Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. [PRMBench: A fine-grained and challenging benchmark for process-level reward models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics.
- 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](#). *arXiv preprint arXiv:2211.14275*.
- Chenglong Wang, Yongyu Mu, Hang Zhou, Yifu Huo, Ziming Zhu, Jiali Zeng, Murun Yang, Bei Li, Xiaoyang Hao, Chunliang Zhang, Fandong Meng, Jingbo Zhu, and Tong Xiao. 2026. [GRAM-R²: Self-training generative foundation reward models for reward reasoning](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33395–33403.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. [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, Bangkok, Thailand. Association for Computational Linguistics.

- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. [Evaluating mathematical reasoning beyond accuracy](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(26):27723–27730.
- Bin Xie, Bingbing Xu, Xueyun Tian, Yilin Chen, and Huawei Shen. 2026. [Towards robust process reward modeling via noise-aware learning](#). *arXiv preprint arXiv:2601.12748*.
- Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. An Implementation of Generative PRM. <https://github.com/RLHFlow/RLHF-Reward-Modeling>.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- 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](#). *arXiv preprint arXiv:2409.12122*.
- Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. 2025b. [Beyond the first error: Process reward models for reflective mathematical reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4711–4728, Suzhou, China. Association for Computational Linguistics.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.
- Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. [ReST-MCTS*: LLM self-training via process reward guided tree search](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 64735–64772. Curran Associates, Inc.
- Jianghangfan Zhang, Yibo Yan, Kening Zheng, Xin Zou, Song Dai, and Xuming Hu. 2026. [GM-PRM: A generative multimodal process reward model for multimodal mathematical reasoning](#). In *Proceedings of the 4th Workshop on Advances in Language and Vision Research (ALVR)*, pages 139–154, San Diego, California, USA. Association for Computational Linguistics.
- 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](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria. Association for Computational Linguistics.
- Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2026. [GenPRM: Scaling test-time compute of process reward models via generative reasoning](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34932–34940.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. [ProcessBench: Identifying process errors in mathematical reasoning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-Bench and chatbot arena](#). *Advances in Neural Information Processing Systems*, 36:46595–46623.
- Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. 2024. [PaD: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2571–2597, Mexico City, Mexico. Association for Computational Linguistics.

A Benchmarks

- ProcessBench (Zheng et al., 2025) evaluates error localization in mathematical reasoning processes. It contains 3,400 test examples drawn from multiple mathematical reasoning datasets. Each example consists of a problem, a step-by-step solution, and a human-annotated error position. If the solution is incorrect, the annotation marks the earliest erroneous step; otherwise, the trajectory is labeled as fully correct. During evaluation, the model must judge whether the full trajectory is correct and, when an error exists, identify the first erroneous step.

- PRMBench (Song et al., 2025) is a fine-grained benchmark for evaluating process reward models. It contains 6,216 problems and 83,456 step-level labels, covering diverse process-level judgment dimensions. Unlike benchmarks focused only on first-error localization, PRMBench evaluates process verification from multiple aspects, including conciseness, reliability, and sensitivity. Its submetrics are Non-Redundancy (NR), Non-Circular Logic (NCL), Empirical Soundness (ES), Step Consistency (SC), Domain Consistency (DC), Confidence Invariance (CI), Prerequisite Sensitivity (PS), Deception Resistance (DR), and Multi-Solution Consistency (MS). It includes both synthetic and human-verified data, with annotation quality controlled by human inspection. We use its official metrics, where PRMScore combines F1 performance on positive and negative samples to provide a balanced measure of correct and erroneous step recognition.

B Implementation Details

B.1 Difficulty-Adaptive Rollout Sampling

To control the global rollout budget, we first use a teacher model to classify problems in the *MATH* dataset into three difficulty levels: hard, medium, and easy. We then assign different sampling budgets and early-stopping criteria to each group. For hard problems, we sample up to 32 trajectories and stop once at least one trajectory reaches the correct final answer. For easy problems, we sample up to 16 trajectories and stop once at least one trajectory produces an incorrect final answer. For medium problems, we sample up to 64 trajectories and stop once at least two correct and two incorrect trajectories are obtained. For each problem, we retain at most four reasoning paths, preferably balanced as two correct and two incorrect trajectories. This strategy avoids redundant rollouts on overly easy or overly difficult problems while ensuring that the resulting process supervision covers both positive verification and first-error localization scenarios.

B.2 Cognitive Scaling Hyperparameters

During cognitive scaling, we apply four rule checks to all steps and invoke QwQ-32B only for steps in the ambiguous MC range, steps flagged by a rule, and candidate pivot steps. The ambiguous range is $MC^{raw} \in [0.2, 0.8]$. The checks cover problem

Check	Operational output and representative trigger
Problem relevance	−0.1 for an explicit off-topic cue; 0 when topical wording alone does not establish correctness. This conservative check does not award +0.1 from lexical overlap.
Variable-reference consistency	−0.1 for an unintroduced variable; 0 for previously introduced variables, generic template variables, variable-free steps, or malformed text that cannot be assessed reliably.
Numeric/equation validity	+0.1 for a safely evaluable equality that holds, −0.1 when it is false, and 0 for symbolic or unsupported expressions that cannot be evaluated safely.
Expression anomaly	−0.1 for explicit malformed text, unmatched delimiters, or literal division by zero; 0 when no such anomaly is detected. Absence of an anomaly is not positive evidence.

Table 4: Operational outputs of the four rule checks. Not every check emits all three values; unverifiable or merely non-violating cases remain neutral.

relevance, variable-reference consistency, safely evaluable numeric/equation validity, and explicit expression anomalies; a check abstains when its condition cannot be evaluated reliably.

Each rule returns +0.1 only for directly verified positive evidence, −0.1 for an explicit violation, and 0 for abstention or non-applicability. Multiple rules may fire and are summed; failure to detect an error is neutral rather than positive evidence. QwQ-32B selects at most one dominant cognitive dimension—information faithfulness and relevance, logical inference, concept application, step coherence, or expression construction—with signed magnitude 1.0, 0.8, 0.6, 0.4, or 0.2, respectively, and returns 0 when local evidence is insufficient. The five dimensions are not accumulated. The total semantic signal therefore lies in $[-1.4, 1.4]$; with $\lambda = 0.5$, the pre-clipping multiplier in Eq. 2 lies in $[0.3, 1.7]$.

Operational order. For each step, the four checks are evaluated independently and their signed outputs are summed. QwQ-32B is then queried only when the routing criteria above are met; it returns one dominant signed cognitive weight, or 0 upon abstention. The rule sum and this single QwQ weight are combined in Eq. 2, multiplied by λ , and clipped to $[0, 1]$. Thus, neither an absent rule violation nor an unselected cognitive dimension contributes positive evidence.

Pseudocode: Step-Cognitive MC Value Scaling

```

1 Input:  $x, s_{<t}, a_t, MC_t^{\text{raw}}$ , and  $\lambda$ 
2  $\text{rule\_sum} \leftarrow 0$ 
3 for each
   $r \in \{\text{relevance, variable, equation, anomaly}\}$  do
4    $w \leftarrow r(x, s_{<t}, a_t)$  ( $w \in \{-0.1, 0, +0.1\}$ )
5    $\text{rule\_sum} \leftarrow \text{rule\_sum} + w$ 
6  $\text{route} \leftarrow (MC_t^{\text{raw}} \in [0.2, 0.8])$  or any nonzero rule
  output or  $\text{candidate\_pivot}(t)$ 
7 if  $\text{route}$  then  $w_{\text{llm}} \leftarrow \text{QwQ}(x, s_{<t}, a_t, MC_t^{\text{raw}})$ 
8    $w_{\text{llm}}$  is one dominant signed weight, or 0 on abstention
9 else  $w_{\text{llm}} \leftarrow 0$ 
10  $V_t \leftarrow \text{clip}(MC_t^{\text{raw}}[1 + \lambda(\text{rule\_sum} + w_{\text{llm}})], 0, 1)$ 
11 return  $V_t$ 

```

B.3 Pivot-Step-Directed Repair and Filtering Hyperparameters

Pivot step localization uses the calibrated step values V_t . We set the reliable-prefix threshold to $\tau_{\text{high}} = 0.6$, the low-value threshold to $\tau_{\text{low}} = 0.1$, and the significant-drop threshold to $\delta_{\text{drop}} = 0.5$. For each trajectory with an incorrect final answer, we select the first step satisfying the pivot condition. For candidate repairs, we require both the stability constraint $V(\hat{s}_{t^*}^{(m)}) \geq 0.5$ and the Sibling Advantage constraint $\text{Adv}(\hat{s}_{t^*}^{(m)}, s_{t^*}) \geq 0.5$. Only candidates satisfying both constraints are retained as golden correction pairs.

The repair and verification budgets are allocated according to the global pass rate

$$P_{\text{global}} = \frac{1}{|\mathcal{P}_x|} \sum_{\gamma \in \mathcal{P}_x} \mathbb{I}(\text{Ans}(\gamma) = y^*). \quad (10)$$

When $P_{\text{global}} < 0.2$, we set $N_{\text{repair}} = 128$ and $K_{\text{verify}} = 128$; when $0.2 \leq P_{\text{global}} < 0.8$, we set $N_{\text{repair}} = 64$ and $K_{\text{verify}} = 64$; when $P_{\text{global}} \geq 0.8$, we set $N_{\text{repair}} = 32$ and $K_{\text{verify}} = 32$. This allocation gives harder problems larger repair and verification budgets, while using smaller budgets for easier problems where high-confidence filtering is sufficient.

B.4 Stage-wise Threshold Sensitivity

For $\lambda = 0.25/0.5/0.75/1.0$, relative label noise falls by 10.2/11.0/10.6/6.7%, and hard-label coverage is 90.10/89.88/89.94/90.10%. The largest reduction among these settings occurs at $\lambda = 0.5$. Narrowing the raw-MC routing interval from $[0.1, 0.9]$ to $[0.4, 0.6]$ changes score-triggered adjudication coverage from 16.89% to 3.79% (11.52% at the default); these percentages measure cost and coverage, not label quality. Varying the positive-path threshold from 0.4 to 0.9 changes outcome-

correct path retention from 86.9% to 68.8%, with 85.8% retained at the default 0.5.

Setting	Locations/Pairs	Overlap/Share
<i>Pivot localization</i>		
Default (.6, .1, .5)	4,112	1.000
$\tau_{\text{high}} = .5$	4,708	.873
$\tau_{\text{high}} = .7$	3,396	.825
$\tau_{\text{low}} = .2$	4,158	.988
$\delta_{\text{drop}} = .4$	4,336	.938
$\delta_{\text{drop}} = .6$	3,791	.914
<i>Sibling acceptance</i> ($\tau_{\text{stable}}, \delta_{\text{adv}}$)		
(.5, .5)	3,541	100.0%
(.5, .6)	3,396	95.9%
(.5, .8)	2,990	84.4%
(.8, .5)	3,377	95.4%
(.8, .6)	3,293	93.0%
(.9, .8)	2,817	79.6%

Table 5: Stage-wise threshold sensitivity. Pivot rows report Jaccard overlap with the default locations; sibling rows post-filter stored default candidates and report the retained share of the default 3,541 pairs.

Across the full 27-setting pivot grid, the step-1–3 share is 47.1–50.4%. The sibling rows use post-filtering rather than end-to-end resampling, and no human repair-quality labels are available. Rule and cognitive weights remain fixed.

B.5 Training Sample Selection

Path-Level Golden Correction Pair. Targeted repair rollout produces path-level golden correction pairs rather than isolated step pairs. Let the original failed trajectory and the repaired trajectory be

$$\begin{aligned} \gamma^- &= (a_1, \dots, a_{t^*}^-, \dots, a_{T^-}^-), \\ \gamma^+ &= (a_1, \dots, a_{t^*}^+, \dots, a_{T^+}^+), \end{aligned} \quad (11)$$

where $a_{t^*}^-$ is the original pivot step, and $a_{t^*}^+ = \hat{a}_{t^*}^{(m^*)}$ is the accepted repaired sibling step. The two trajectories share the reliable prefix $C_{t^*-1} = [x, a_1, \dots, a_{t^*-1}]$ and first diverge at step t^* . We define the path-level golden correction pair as

$$\mathcal{G} = (x, y^*, C_{t^*-1}, B^-, B^+, t^*), \quad (12)$$

where $B^- = (a_{t^*}^-, \dots, a_{T^-}^-)$, $B^+ = (a_{t^*}^+, \dots, a_{T^+}^+)$, and the pair satisfies $\text{Ans}(\gamma^-) \neq y^*$ and $\text{Ans}(\gamma^+) = y^*$.

The two paths share C_{t^*-1} and first differ at t^* . The pair $(a_{t^*}^-, a_{t^*}^+)$ records the replacement under fixed context, while (B^-, B^+) records the downstream outcomes. Acceptance is based on the final outcome and does not certify every later step.

Sample Selection. For each golden correction pair $\mathcal{G}$, we construct repair-aware samples from its

shared reliable prefix, original branch, and repaired branch. For a trajectory $\gamma = (a_1, \dots, a_T)$ with a correct final answer, we include it as a positive verification sample if all steps satisfy $V_t \geq \tau_{\text{pos}}$. For a failed trajectory, we define the first low-value step as

$$e(\gamma) = \min\{t \mid V_t \leq \tau_{\text{neg}}\}, \quad (13)$$

and require all preceding steps to satisfy $V_i \geq \tau_{\text{pos}}$. Trajectories satisfying this condition are used as negative verification samples. Repair-aware samples are retained only when the repaired branch is accepted by the stability and Sibling Advantage filters in Eq. 6.

B.6 Training Configuration

CePRM is fine-tuned with LoRA rank 32 and alpha 64 for three epochs. We use a maximum sequence length of 6,144, a per-device batch size of 1, gradient accumulation over 32 steps, and a learning rate of 1×10^{-4} with cosine decay and a warmup ratio of 0.03. The weight decay is 0.01. Training uses BF16, gradient checkpointing, ZeRO-2, and random seed 42. The experiments were run on four NVIDIA GeForce RTX 4090 GPUs.

B.7 Generation and Parsing Settings

Stage	Model	Decoding or budget
Synthesis	QwQ-32B	$T = 0.7$, $\text{top-}p = 0.9$; max 3,072 tokens
ProcessBench	CePRM-4B	$T = 0$, $\text{top-}p = 1$; max 2,048 new tokens
Best-of-8	CePRM-4B	$M = 1$; max 1,024 new tokens

Table 6: Generation settings used for rationale synthesis and author-evaluated inference. T denotes temperature.

Rationale synthesis returns nonempty <analyze> and <fix> fields, a boxed Yes/No judgment per step, and a 1-based first-error index. Deterministic filtering removes malformed outputs, label disagreements, and repair samples without the required repair field. ProcessBench retains prior verification turns; Best-of-8 uses the minimum step reward per path (Appendix C).

C Best-of-8 Reranking

Table 7 evaluates downstream reranking on 959 questions across six math benchmarks. A Qwen3-4B policy generates a shared $K = 8$ pool (temperature 1.0, top- p 1.0, 2,048 tokens, seed 20260710). Pass@1 averages correctness across sampling positions, and Pass@8 is oracle coverage. Maj@8

Benchmark	Policy		CePRM	Oracle
	Pass@1	Maj@8	4B	Pass@8
AIME24	8.33	13.33	16.67	20.00
AMC23	41.25	50.00	52.50	60.00
MATH500-clean	37.96	44.24	47.00	58.06
OlympiadBench	33.69	40.50	42.00	47.50
College Math	30.13	32.00	32.50	33.00
Minerva Math	29.18	33.46	35.66	44.85
Macro average	30.09	35.59	37.72	43.90

Table 7: Best-of-8 reranking accuracy (%) on a shared Qwen3-4B candidate pool. Pass@8 is oracle candidate-pool coverage. MATH500-clean excludes 283 exact question matches found against the 21,421/1,000 SFT train/validation conversations, leaving no exact question overlap with the evaluation set.

votes over normalized nonempty answers, retains duplicates, and breaks ties by first occurrence; all-empty pools default to the first candidate. CePRM selects the first path with maximal minimum step reward and assigns 0.5 to a missing verifier output. No paired confidence interval was computed.

D Error-Position, Length, and Repair-Scope Diagnostics

On ProcessBench, first-error accuracy is 79.1/73.3/68.3% for positions 1–3/4–6/7+. For lengths $\leq 5/6$ –10/11+, error-side accuracy is 75.9/74.8/69.9%, and correct-side accuracy is 85.3/76.7/61.0%; these results therefore do not establish deep-step or long-chain generalization.

E Prompt Templates and Deterministic Filtering

The following tables reproduce the prompts used in data construction and evaluation. Braced fields are replaced at runtime.

Component	Prompt or runtime content
Difficulty system	You are a helpful assistant that evaluates math problem difficulty.
Difficulty user	You are an expert math teacher. Evaluate the difficulty of the following math problem for a high school or undergraduate student. Problem: {problem} Determine the difficulty level from these three categories: 1. Easy: Basic application of concepts, single-step or simple multi-step. 2. Medium: Requires combining multiple concepts, non-trivial logic, or moderate calculation. 3. Hard: Competition level (e.g., AIME, USAMO), complex proofs, or very tricky logic. Provide a JSON response with a single field: "difficulty" (Easy, Medium, or Hard). Example: {"difficulty": "Medium"}
Rollout/continuation system	Please reason step by step, and put your final answer within \boxed{ }.
User	{problem}
Initial assistant prefix	Step 1:
Prefix-MC assistant prefix	{Step 1: a_1} ... {Step t: a_t} Step {t+1}:
Gold-answer access	The gold answer is not provided during generation; it is used only afterward by the answer extractor to determine terminal correctness. The historical template contains no additional instruction prohibiting revision of the supplied prefix.

Table 8: Difficulty-classification and rollout/prefix-continuation prompts used in data construction.

Component	Cognitive-adjudication prompt
Task and inputs	You are a mathematical-process adjudicator. Assess only the current step under the supplied problem and prefix. Do not solve the entire problem, use the gold answer, reward style, or infer correctness solely from a correct final answer. Problem: {problem} Reliable prefix (possibly empty): {prefix} Current step: {step} Raw MC value (selection metadata; do not treat as mathematical evidence): {raw_mc}
Dimension selection	Choose at most one dominant dimension, only when there is explicit evidence. The selected dimension fixes the absolute weight; do not choose or combine magnitudes independently: 1. information_faithfulness_and_relevance: 1.0 2. logical_inference: 0.8 3. concept_application: 0.6 4. step_coherence: 0.4 5. expression_construction: 0.2
Sign and abstention	Use the positive sign when the current step explicitly demonstrates the selected capability and the negative sign when it explicitly violates it. Use dimension none and weight 0.0 when local evidence is insufficient. Never return more than one dimension or sum dimension weights. The weight is an impact-coded adjustment, not a probability. Do not issue a positive weight merely because the wording is fluent or the prefix is plausible.
JSON-only output	<pre>{ "dominant_dimension": "one of the five names above, or none", "signed_weight": 0.0, "evidence_span": "a short exact span from the current step", "reason": "one concise local mathematical justification", "abstained": true }</pre>

Table 9: Cognitive-adjudication prompt. Raw MC is routing metadata rather than mathematical evidence, and the adjudicator returns at most one signed dimension weight.

Component	Prompt or acceptance rule
Diagnosis system	You are a helpful assistant that outputs JSON.
Diagnosis user	You are a professional math teacher. Please evaluate the student’s problem-solving process. Question: {problem} Previous Correct Steps: {context_steps_or_None} Current Incorrect Step: {current_step} Instructions: 1. Analyze why the ‘Current Incorrect Step’ is wrong based on the ‘Question’ and ‘Previous Correct Steps’. 2. Classify the error into EXACTLY ONE of the following types: - Logical Error: The logical reasoning or deduction is incorrect. - Expression Error: The mathematical expression is written incorrectly, or symbols are misused. - Calculation Error: The arithmetic calculation is wrong. - Concept Error: A mathematical concept, theorem, or formula is misunderstood or misapplied. - Missing Step: A crucial intermediate step is skipped, making the transition invalid. 3. Output your response STRICTLY in the following JSON format: <pre>{“teacher_analysis”: “<analyze>Your detailed explanation of the error.</analyze>”, “error_type”: “One of the 5 exact types listed above”}</pre>
Directed-repair user	You are a mathematical problem solver. Please continue solving the problem. Question: {problem} Previous Correct Steps: {context_steps_or_None} We previously attempted the following step: {error_step} However, this step contains a {error_type}. Teacher’s analysis: {teacher_analysis} Please provide ONE corrected step to replace it, and then continue solving the rest of the problem until you reach the final answer.
Directed-repair context	The directed-repair call uses the rollout system instruction in Table 8 and supplies the reliable prefix as the assistant prefix.
Candidate acceptance	A candidate is retained only if its terminal answer is correct, its replacement step differs from the original erroneous step, it passes the stability and Sibling Advantage gates, and it has the maximum advantage under Eq. 6.

Table 10: Error-type diagnosis and directed-repair prompts, followed by the deterministic candidate-acceptance conditions.

Prompt Component	Content
System	You are a math teacher. Your task is to review and critique the solution step by step.
Role and Task	You are a Math Teacher. Review the following complete solution path step by step. For each step i , judge correctness, explain the reason, and provide correction instructions in the <code><fix></code> tag.
Problem Input	[Problem] {problem}
Full Solution Path	[Full Solution Steps] Step 1: {step_1} Step 2: {step_2} ... Step n: {step_n}
Optional Reference Repair Pair	If a repair pair is available, we additionally provide the shared context, the incorrect step, and its repaired counterpart: [Reference Repair Pair Step k] [Context] Step 1: {context_1} ... [Incorrect Step] {bad_step} [Correct Step] {good_step}
Correction Rules	Correct step: output <code><fix>No correction needed.</fix></code> . Incorrect step with reference repair: use the provided [Correct Step] to give a concrete rewritten mathematical step. Example: <code><fix>The calculation is wrong. Correct step: [Insert content from reference]</fix></code> Incorrect step without reference repair: do not rewrite the math; describe the error type and state that the branch should be discarded. Example: <code><fix>This step contains a logical/calculation error. Since no valid repair path is available, this branch should be discarded.</fix></code>
Cascading Failure Rule	If the immediately preceding step was judged No, all subsequent steps must follow: <code><analyze>State that since the previous step [Index] was incorrect, the current step derives from a wrong premise and cannot be validated.</analyze></code> <code><fix>Stop correction. This path is invalid due to prior errors.</fix></code> <code><output>**Judgement**<output></code>
Required Output Format	For each step i , strictly output: Step i <code><analyze>...</analyze></code> <code><fix>...</fix></code> <code><output>**Judgement**<output></code> After all steps, output the earliest wrong step index, or -1 if no error exists: <code><output>\$\boxed{INDEX}\$</output></code> The index is 1-based rather than 0-based.

Table 11: Rationale-synthesis prompt used to construct generative process-supervision data. The source-code typo “subsequent” is corrected to “subsequent” without changing the instruction.

Component	Rule or prompt content
Consensus filtering	There is no independent consensus-filtering prompt or second QwQ call. The deterministic parser requires nonempty <analyze> and <fix> fields; positive verification must yield Yes, while negative or critique samples must yield No. For a repair critique with a reference step, the repair field must also be nonempty and pass the corresponding check. Any format, label, or repair-consistency failure is discarded.
ProcessBench/BoN system	You are a math teacher. Your task is to review and critique the solution step by step. For the current step, respond in the following format: <analyze>...</analyze> <fix>...</fix> <output>**Judgement**:$\boxed{\text{Yes No}}$</output> Use $\boxed{\text{Yes}}$ if the step is correct, otherwise $\boxed{\text{No}}$.
First user turn	Question: {problem} {current_step_or_Step_1_prefix}
Conversation construction	Each later step is appended as a new user turn while all prior user/assistant verification turns are retained. ProcessBench uses temperature 0, top- $p = 1$, and at most 2,048 new tokens. Best-of-8 applies the same verifier step by step with $M = 1$.

Table 12: Deterministic consensus filtering and the shared ProcessBench/Best-of-8 verifier prompt.

Component	PRMBench verifier content
System	You are a math teacher. Your task is to verify the logic and correctness of the Current Step based on the Question and Previous Steps. Perform an independent calculation and analysis. Do not look for or assume the existence of a reference solution. You must respond in this exact format: <analyze>...</analyze> <fix>...</fix> <output>**Judgement**:$\boxed{\text{Yes/No}}$</output> If the step is correct, use $\boxed{\text{No}}$ correction needed.</fix>.
First user turn	Question: {question} Step 1: {step_1}
Later user turn	Step {k}: {step_k}
Fixed prior assistant history	<analyze>The step is logically sound and the calculation is correct. No errors detected.</analyze> <fix>No correction needed.</fix> <output>**Judgement**:$\boxed{\text{Yes}}$</output>
Reference access	When step k is evaluated, the fixed positive assistant history is inserted for the preceding $k - 1$ steps. No reference solution is supplied to the model.

Table 13: CePRM system prompt and multi-turn input template used for PRMBench evaluation.