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

Anonymous ACL submission

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, 2024a; Anthropic, 2023; OpenAI, 2024b; Guo 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., 2025a; 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 rollout data possesses implicit tree structure that standard pipelines discard by flattening trajectories into independent sequences (Yao et al., 2023; Zhang et al., 2024). In many failed solutions, reasoning remains sound until a single error (Luo et al., 2024; Yang et al., 2025b) triggers a sharp drop in step value, a pattern we term a *value cliff*. 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). The step that triggers the first cliff, the *pivot step*, provides a natural anchor for same-prefix contrastive supervision.

Motivated by these observations, we propose **CePRM**, a generative PRM whose training supervision addresses both MC label noise and under-exploited 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 cliff in each failed trajectory, then samples and verifies corrective continuations under the same reliable prefix. Together, these mechanisms produce

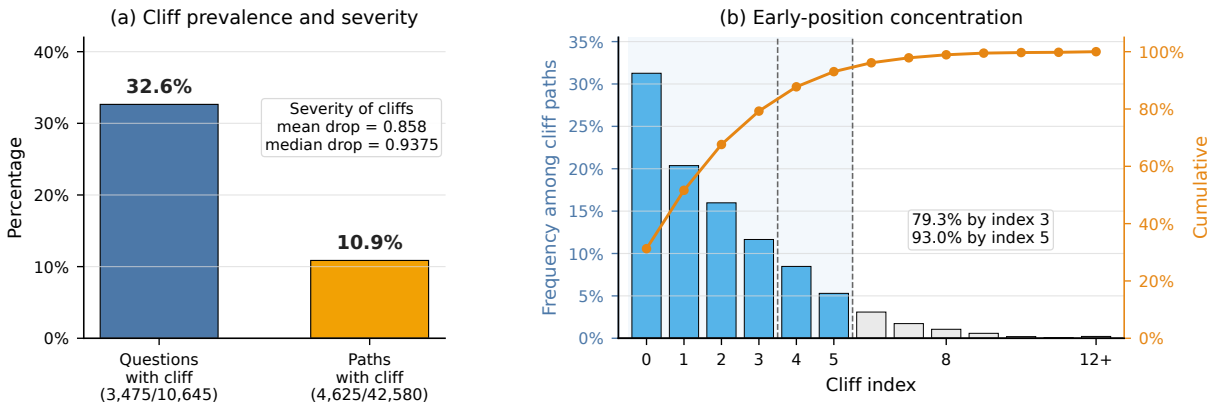

Figure 1: Value cliffs in MC rollout trees. (a) In our rollout pool, 32.6% of questions (3,475/10,645) and 10.9% of paths (4,625/42,580) exhibit a value cliff, with a mean (median) severity of 0.858 (0.938). (b) Cliff positions concentrate early: 79.3% occur by step index 3 and 93.0% by step index 5,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

structured supervision spanning correct, erroneous, and repair-aware verification instances.

Our contributions are as follows:

- We propose CePRM, a generative PRM that unifies semantic-calibrated MC estimation and rollout-tree topology exploitation within a single supervision-construction framework, jointly addressing label noise and the absence of repair-oriented training signal.
- We identify value cliffs as an empirically prevalent and reliable structural signal in rollout trees for locating the first reasoning failure, and introduce pivot-step-directed repair mining with Sibling Advantage filtering to extract verified same-prefix correction pairs.
- With only a 4B backbone, CePRM 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 (Skywork o1 Team, 2024; Luo et al., 2024; Zhang et al., 2025b). 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; subsequent studies extend this paradigm to long-form and self-reflective settings (She et al., 2025; Zhang et al., 2025a; Wang et al., 2026; Guan et al., 2025). These efforts advance PRM architecture but leave supervision quality unaddressed. CePRM targets this axis by calibrating MC values with step-level semantic signals and exploiting value cliffs in rollout trees to mine same-prefix repair contrasts.

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 annota-

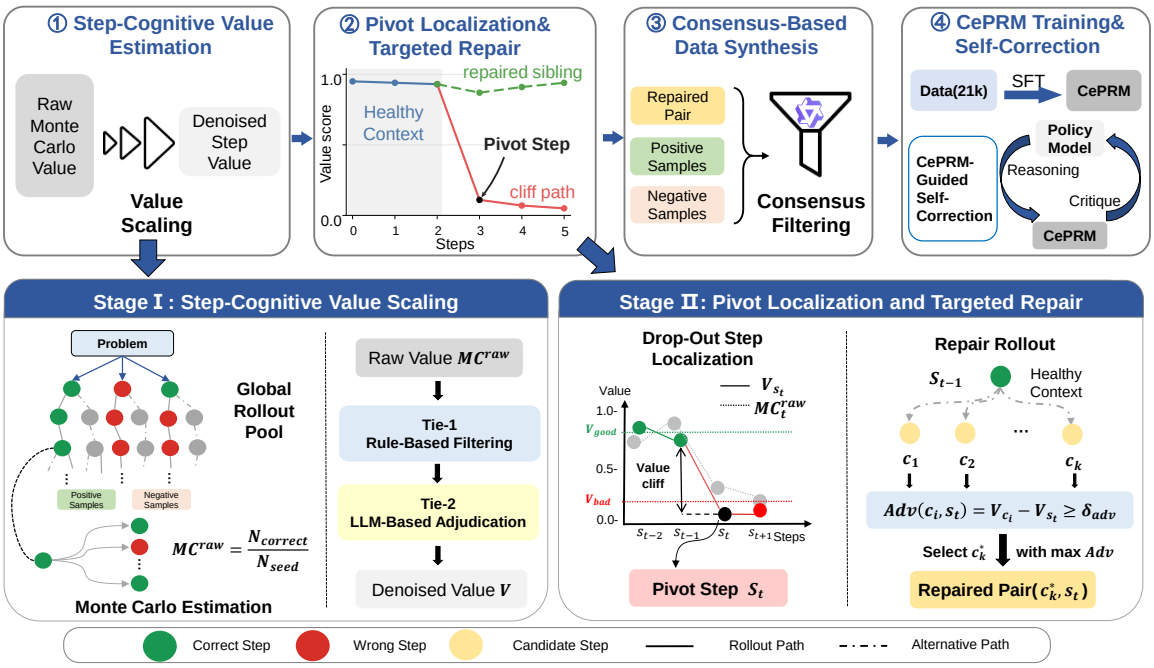

Figure 2: Overview of the CePRM pipeline. Step-Cognitive Value Scaling calibrates raw MC scores through rule-based filtering and LLM-based adjudication to produce denoised step values. Pivot localization and targeted repair identifies value cliffs in failed trajectories and constructs same-prefix correction pairs through Sibling Advantage filtering. Please refer to Section 3 for details.

tions, 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 $\gamma = (a_1, \ldots, a_T)$, we denote the prefix state after step t as $s_t = [x, a_1, \ldots, a_t]$ and compute the raw MC score by sampling K continuations:

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

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.

Step-Cognitive MC Value Scaling addresses this noise by calibrating MC_t^{raw} with step-level semantic signals, producing a value V_t that serves as the basis for all downstream pivot-step localization, selection, and filtering.

Semantic Assessment: Calibration proceeds in two stages, both conditioned solely on raw MC scores. A rule-based layer first screens every step for structural signals: problem relevance, variable consistency, symbolic validity, and abnormal equations. Each rule j produces a signed weight $\omega_{t,j}^{rule}$ indicating whether the step passes or violates the corresponding check. An LLM-based adjudicator is then invoked only for high-impact positions identified from MC_t^{raw}-steps in the ambiguous score range, rule-flagged steps, and steps exhibiting a large adjacent raw-score drop (candidate pivot positions). For each selected step, the adjudicator assigns a signed cognitive weight ω_t^{llm} reflecting the dominant mathematical capability the step exercises or violates; for unselected steps, $\omega_t^{llm} = 0$. Details are provided in Appendix B.2.

Cognitive Scaling Function: Given these signals, the calibrated step value is

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

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 ($V = MC^{raw} + \delta$): *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 a 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*. Value cliffs appear in 32.6% of problems and concentrate in early reasoning steps (Figure 1), making them a reliable signal for locating first failures.

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 $\gamma = (a_1, \ldots, a_T)$, we define the adjacent value drop $\Delta_t = V_{t-1} - V_t$ and select the pivot step by

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

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 such step, the most likely site of the first error, before propagation corrupts downstream values. Trajectories with no qualifying step are discarded for lacking a reliable prefix.

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

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

Each candidate forms a sibling prefix $\hat{s}_{t^\star}^{(m)} = [c, \hat{a}_{t^\star}^{(m)}]$ of the prefix $s_{t^\star} = [c, a_{t^\star}]$. 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:

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

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

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

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:

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

Path-level definitions of the resulting correction pairs appear in Appendix B.4. 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 $\gamma = (a_1, \ldots, a_T)$, 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, a verified repaired branch, and the pivot index $t^\star$ (formal definition in Appendix B.4).

Rationale Synthesis: Prior work shows that explicit rationales improve verification accuracy (Zhu et al., 2024; Zhao et al., 2026). We use a teacher model (Qwen Team, 2025) generates a structured rationale $z_t = \langle \texttt{analyze}, \texttt{fix}, \texttt{output} \rangle$ for each step a_t, conditioned on the problem x and prefix $(a_1, \ldots, a_{t-1})$. 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 produce judgments inconsistent with process labels (Zheng et al., 2023; Zhao et al., 2026); consensus filtering discards such samples. 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 $\mathbb{G}$, $\tilde{e} = t^\star$ with the additional requirement that the teacher's suggested fix reaches the correct final answer under the same prefix as the accepted repaired branch.

Training: The filtered supervision comprises three partitions:

$$\mathcal{D} = \mathcal{D}_{pos} \cup \mathcal{D}_{neg} \cup \mathcal{D}_{rep}. \tag{9}$$

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

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. Baselines span both discriminative PRMs (*Math-Shepherd* (Wang et al., 2024), *RL-HFlow* (Xiong et al., 2024), *Skywork-PRM* (Skywork o1 Team, 2024), *ReasonEval-7B* (Xia et al., 2025), *Qwen2.5-Math-PRM* (Zheng et al., 2025; Zhang et al., 2025b)) and generative PRMs (*R-PRM* (Xiong et al., 2024), *GenPRM* (Zhao et al., 2026)). All comparisons follow the official evaluation protocols; for models with published results we report the corresponding numbers, and otherwise evaluate under the same setting.

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}. The resulting trajectories are processed through Step-Cognitive MC Value Scaling, pivot-step localization, targeted repair mining, and consensus filtering with QwQ (Qwen Team, 2025) as the teacher model, yielding approximately 21K training instances across positive, negative, and repair-aware supervision. All hyperparameters and prompts are listed in Appendix B.

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

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). The improvement is driven by error-side recognition: CePRM-4B attains the highest error-side accuracy among all PRMs on every subset (75.0, 76.0, 71.2, and 72.0 on GSM8K, MATH, OlympiadBench, and OmniMath), with the largest margins on the two harder benchmarks where longer reasoning chains make first-error localization more difficult. 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).

Model	GSM8K			MATH			OLYMPIADBENCH			OMNIMATH			Avg. F1
	err.	corr.	F1	err.	corr.	F1	err.	corr.	F1	err.	corr.	F1	
LLM-as-judge, proprietary models													
GPT-4o*	70.0	91.2	79.2	54.4	76.6	63.6	45.8	58.4	51.4	45.2	65.6	53.5	61.9
o1-mini*	88.9	97.9	93.2	83.5	95.1	88.9	80.2	95.6	87.2	74.8	91.7	82.4	87.9
LLM-as-judge, open-source models													
Qwen2.5-Math-72B-Instruct*	49.8	96.9	65.8	36.0	94.3	52.1	19.5	97.3	32.5	19.0	96.3	31.7	45.5
Qwen2.5-72B-Instruct*	62.8	96.9	76.2	46.3	93.1	61.8	38.7	92.6	54.6	36.6	90.9	52.2	61.2
PRMs													
Math-Shepherd-7B*	32.4	91.7	47.9	18.0	82.0	29.5	15.0	71.1	24.8	14.2	73.0	23.8	31.5
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. Results marked with * come from She et al. 2025. Bold text denotes the best results within PRM.The best results among PRMs are highlighted in **bold**.

Model	Simplicity			Soundness					Sensitivity				Overall
	NR.	NCL.	Avg.	ES.	SC.	DC.	CI.	Avg.	PS.	DR.	MS.	Avg.	
LLM-as-judge, proprietary models													
GPT-4o*	57.0	62.4	59.7	72.0	69.7	70.7	71.1	70.9	62.5	65.7	99.2	75.8	66.8
o1-mini*	65.6	63.7	64.6	74.5	67.7	73.8	72.3	72.1	61.8	64.8	100.0	75.5	68.8
PRMs													
Math-Shepherd-7B*	44.0	50.3	47.1	49.4	44.5	41.3	47.7	45.7	47.2	48.6	86.1	60.7	47.0
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. Results marked with * come from She et al. 2025. Bold text denotes the best results within PRM.The best results among PRMs are highlighted in **bold**.

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.

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%). This asymmetry follows from the multiplicative form of Eq. 2: semantic signals can suppress overestimated values but cannot override uniformly

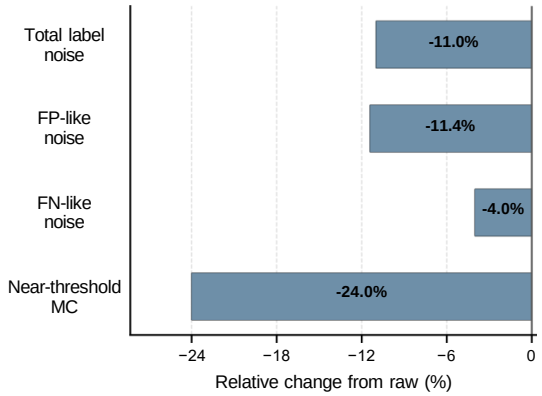

Figure 3: Effect of Step-Cognitive MC Value Scaling on MC label noise.

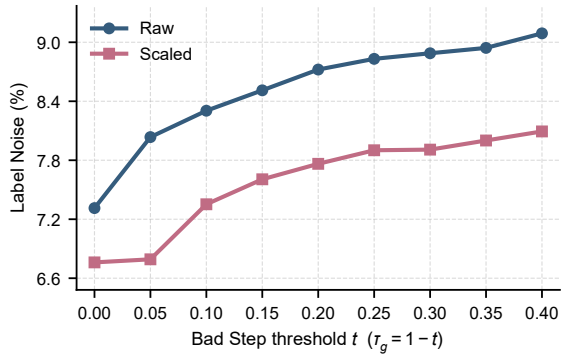

Figure 4: Sensitivity of label noise to the bad-step threshold t, with $\tau_g = 1 - t$.

weak rollout evidence.

The improvement is robust to threshold choice. Figure 4 sweeps $\tau_{\text{neg}} \in [0, 0.4]$ with $\tau_{\text{pos}} = 1 - \tau_{\text{neg}}$. Although noise increases for both raw and scaled labels as τ_{neg} grows, the scaled curve remains consistently below the raw curve, confirming that denoising is not an artifact of a single operating point. 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.

4.4 Ablation Study

Table 3 isolates the contribution of each CePRM component: generative verification rationales (CoT), Step-Cognitive MC Value Scaling (VS), and pivot-step-directed repair mining (DR).

Generative rationales establish a strong baseline. Switching from discriminative to generative verification raises average F1 from 56.8 to 65.6 (+8.8), confirming prior findings that explicit step-level rationales benefit process reward modeling (Zhao et al., 2026). We treat this generative CoT setting as the reference baseline for all subsequent comparisons.

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.

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 in regions 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. Adding DR to the CoT baseline raises average F1 to 68.4, with the largest gain on OmniMath (+5.0), confirming that same-prefix contrastive corrections supply supervision beyond what standard verification rationales provide. 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). Notably, the marginal contribution of VS is larger when combined with DR (+5.3) than in the discriminative setting alone (+3.4), indicating that calibrated values benefit not only label quality but also the pivot localization and sibling filtering that underpin repair mining. These results confirm that VS and DR address complementary axes of supervision: VS refines the MC-derived values governing selection and filtering, while DR provides locally contrastive correction pairs that teach the model to identify and repair erroneous

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.

reasoning steps.

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, 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

Due to computational constraints, our experiments are conducted primarily on 4B-scale models. We have not yet systematically evaluated the effectiveness and scalability of CePRM on larger models, such as 70B-scale models. Whether CePRM yields consistent gains as model scale increases remains to be further examined. Additionally, we use the constructed repair samples mainly through supervised fine-tuning, although these samples naturally encode local preference relations between valid repairs and erroneous continuations. Future work may further incorporate reinforcement learning, DPO, or other preference-optimization methods to better exploit these relative advantage signals and improve the error localization and correction capabilities of generative PRMs.

References

Anthropic. 2023. Introducing Claude. https://www.anthropic.com/index/introducing-claude/.

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. *arXiv preprint arXiv:2501.04519*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. *arXiv preprint arXiv:2103.03874*.

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, and 1 others. 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. 2024a. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms.

OpenAI. 2024b. 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, YK Li, Yang Wu, and 1 others. 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 13449–13462.

Skywork o1 Team. 2024. Skywork-o1 Open Series. https://huggingface.co/Skywork.

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.

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, and 1 others. 2026. Gram-r^2: Self-training generative foundation reward models for reward reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 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, and 1 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, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Zhaohui Yang, Chenghua He, Xiaowen Shi, Linjing Li, Qiyue Yin, Shihong Deng, and Daxin Jiang. 2025b. Beyond the first error: Process reward models for reflective mathematical reasoning. *arXiv preprint arXiv:2505.14391*.

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. *Advances in neural information processing systems*, 36:11809–11822.

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. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Jianghangfan Zhang, Yibo Yan, Kening Zheng, Xin Zou, Song Dai, and Xuming Hu. 2025a. Gm-prm: A generative multimodal process reward model for multimodal mathematical reasoning. *arXiv preprint arXiv:2508.04088*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and 1 others. 2026. Genprm: Scaling test-time compute of process reward models via generative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 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.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 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.

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. 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 lightweight rule-based detection to all steps and invoke the LLM-based semantic adjudicator only for steps in the ambiguous MC range, rule-flagged abnormal steps, and candidate pivot steps. The ambiguous range is defined as $MC^{raw} \in [0.2, 0.8]$. The rule-based layer detects problem relevance, variable-reference consistency, symbolic and equation validity, and obvious expression anomalies. Each rule signal is assigned a lightweight calibration weight of ± 0.1. The LLM adjudicator uses five symmetric cognitive dimensions: information faithfulness and relevance, logical inference, concept application, step coherence, and expression construction, with absolute weights $1.0, 0.8, 0.6, 0.4, 0.2$, respectively. The scaling coefficient is set to $\lambda = 0.5$.

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 $V(\hat{s}_{t^\star}^{(m)}) \geq 0.5$ and the Sibling Advantage constraint $\text{Adv}(\hat{s}_{t^\star}^{(m)}, s_{t^\star}) \geq 0.5$. Only candidates satisfying both constraints are retained as golden correction pairs.

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

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

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 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

$$\begin{aligned} \gamma^- &= (a_1, \ldots, a_{t^\star}^-, \ldots, a_{T^-}^-), \\ \gamma^+ &= (a_1, \ldots, a_{t^\star}^+, \ldots, a_{T^+}^+), \end{aligned} \tag{11}$$

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

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

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

This structure provides both local contrast and path-level verifiability. The pair $(a_{t^\star}^-, a_{t^\star}^+)$ isolates the difference between the pivot step and an accepted sibling under the same context, while (B^-, B^+) records the corresponding downstream branches. Thus, pivot-step-directed repair mining converts linear trajectory labels into same-prefix sibling contrasts for repair-aware supervision.

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 $\gamma = (a_1, \ldots, a_T)$ 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

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

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.

C Prompt Template for Data Construction

Prompt Component	Content
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 `<fix>` 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 `<fix>No correction needed.</fix>`. **Incorrect step with reference repair:** use the provided `[Correct Step]` to give a concrete rewritten mathematical step. **Incorrect step without reference repair:** do not rewrite the math; describe the error type and state that the branch should be discarded.
Cascading Failure Rule	If the immediately preceding step was judged incorrect, all subsequent steps must be marked incorrect because they derive from a wrong premise. The model should stop correction and state that the path is invalid due to prior errors.

Required Output Format	For each step i, strictly output:
	
	`Step i`
	`<analyze>...</analyze>`
	`<fix>...</fix>`
	`<output>**Judgement**: $\boxed{Yes/No}$</output>`
	
	After all steps, output the earliest wrong step index, or -1 if no error exists:
	`<output>$\boxed{INDEX}$</output>`
	The index is 1-based rather than 0-based.

Table 4: The Prompt Template to Construct Data