SPRM: From Cooperative Games to Marginal-Contribution Process Reward Modeling

Yu Bao[1,2]* **Pak Lon Ip**[1]* **Qiyu Ruan**[1,2] **Xitong Gao**[3]† **Chengzhong Xu**[1,2]†

[1] SKL of Internet of Things for Smart City, University of Macau.
[2] Department of Computer Science, University of Macau.
[3] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences.

Abstract

Process reward models provide step-level feedback for reasoning, but reliable process supervision remains difficult to obtain: human annotations are costly, Monte Carlo rollouts require large sampling budgets, and LLM judges can introduce prompt-sensitive biases. We introduce SPRM, an outcome-only framework that constructs process rewards directly from terminal correctness signals. SPRM views a reasoning trajectory as a temporally ordered cooperative game, where reasoning steps act as participants and the final outcome reward is the shared payoff. Following this allocation view, SPRM learns a differentiable trajectory value function from outcomes and distributes the baseline-relative payoff difference to steps through Aumann-Shapley-style Step Value Integration. We further analyze its Shapley properties under the temporal and causal structure of reasoning trajectories. To stabilize supervision, SPRM incorporates model-counterfactual consistency and estimates a cross-trajectory inherent score that captures each step's context-independent value from semantically similar steps. These signals train a dual-head process reward model that jointly represents trajectory-specific marginal credit and cross-trajectory inherent step value. Across ProcessBench, PRMBench, and Best-of-N reranking on AceMath-RewardBench, SPRM improves first-error localization, marginal-contribution-sensitive evaluation, and test-time reasoning selection over existing open PRMs trained from human, rollout, curated, or outcome-broadcast supervision. These results suggest that marginal-contribution-based credit redistribution offers an effective and scalable route to process supervision from outcome-only data. Our code is available at `https://github.com/by0214/sprm-open`.

1 Introduction

Large language models (LLMs) have demonstrated strong reasoning ability on tasks such as mathematical proof generation [3, 9, 13, 35], interactive decision making [22], and code synthesis [26, 24]. Yet their training and alignment are still dominated by outcome-level signals: rewards are typically assigned only to the final answer, while intermediate steps in reasoning chains receive little direct credit. This sparse-reward setting creates a severe temporal credit-assignment problem [25, 5, 10]. A model can observe whether a trajectory eventually succeeds, but it cannot directly tell which steps contributed to that success. Classical reinforcement learning methods such as PPO [28] and reasoning-oriented variants such as GRPO [29] therefore still optimize mainly from outcome-level rewards, while reward-free preference optimization methods such as DPO [27] learn from trajectory-level preferences over final outcomes. Neither paradigm by itself identifies which intermediate reasoning

*Equal contribution.
†Corresponding authors: `xt.gao@siat.ac.cn` and `czxu@um.edu.mo`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

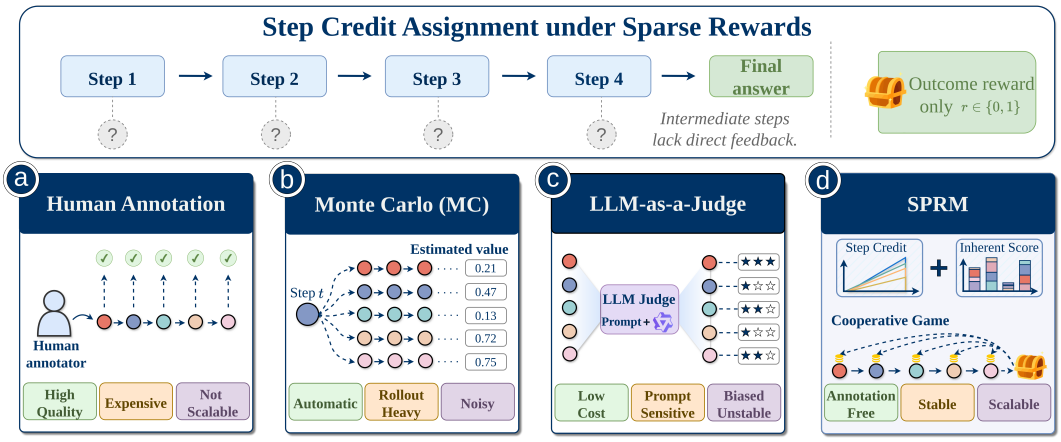


Figure 1: Comparison of PRM supervision under sparse outcome rewards. (a) Human annotation is reliable but costly, while (b) rollout-based estimation and (c) LLM judging provide noisy and unstable supervision. Unlike these external estimators, SPRM derives step-level supervision from outcomes through a cooperative-game view.

steps deserve credit. As a result, process reward models (PRMs) [38, 21, 18], which learn step-level reward signals for reasoning processes, have become an important component in test-time scaling, reranking, and reinforcement learning for LLM reasoning.

The main challenge in developing current process reward models is not the model architecture, but obtaining reliable training data for assigning credit to individual reasoning steps. Human annotation [34, 20] is reliable but expensive and difficult to scale. Monte Carlo-based labeling [36, 44] is more automatic and therefore less expensive, but it requires large rollout budgets and often remains noisy, since the same step can receive different labels across sampled continuations and value estimates. LLM-as-a-judge [42, 14, 40, 37, 4] further reduces annotation cost, yet its labels can inherit biases from pretraining data and be sensitive to prompt wording or scoring criteria, which makes the resulting credits unstable. As illustrated in Figure 1, these approaches share a common mismatch: they import external annotation, rollout, or judge distributions to explain trajectories whose only observed training signal is the final outcome. This dependence limits robustness, scalability, and reusability, and motivates the question of whether reliable step credit can be derived directly from trajectory data itself.

Beyond the dependence on external supervision, we argue that a good step-level credit assignment should also account for each step's marginal contribution to the final trajectory reward. In a successful reasoning chain, a decisive transformation, a harmless restatement, and an irrelevant detour are all paired with the same terminal reward, but they should clearly receive different credit. A contribution-faithful signal should therefore concentrate the final reward on the steps that actually support the outcome, assign little credit to steps whose presence does not change the result, and avoid changing its judgment merely because reasoning steps appear in different positions. We therefore formulate the reasoning process as a cooperative game in which steps participate in the allocation while retaining their temporal dependencies: each step is a participant, the terminal reward is the shared payoff, and credit should be determined by contribution rather than temporal position. From this perspective, sparse outcome supervision becomes a question of payoff allocation: how should a terminal reward be redistributed into principled step-level credit?

Shapley values provide a principled starting point because they allocate payoff by marginal contribution, making them well matched to the goal of converting a sparse final reward into step-level credit. However, reasoning trajectories differ from the classical Shapley setting in a crucial way: steps are temporally ordered, causally dependent, and not freely exchangeable. Under these constraints, a naive Shapley construction can degenerate into prefix-wise value differences along the original sequence, making the resulting credit highly sensitive to the context in which a step appears and often still dependent on strong external value estimators or evaluators. The central challenge is therefore to obtain Shapley-style, contribution-faithful credit from terminal outcome signals alone while respecting the temporal-causal structure of reasoning.

Motivated by this goal, we propose **SPRM** (**S**hapley-inspired **P**rocess **R**eward **M**odel), a framework that automatically constructs PRM training signals from outcome-only rewards. SPRM uses the idea of Aumann-Shapley, a continuous-space variant of Shapley values, to perform marginal-contribution-based credit allocation over reasoning steps. Combined with model-counterfactual consistency, this allocation captures how a step changes the value of its own reasoning trajectory while accounting for interactions among reasoning steps. It further uses a cross-trajectory inherent score to improve the stability of PRM supervision for steps across different contexts.

SPRM directly addresses a central limitation of existing PRM data-construction pipelines: obtaining reliable step-level supervision without costly human annotation or strong external evaluators. SPRM instead extracts PRM supervision directly from sparse terminal rewards, thereby removing dependence on strong external estimators. This provides an automated and scalable route to process supervision while preserving the temporal grounding needed for multi-step reasoning.

Our main contributions are summarized as follows:

- We introduce an outcome-only PRM data-construction framework that derives step-level training labels from terminal correctness rewards, without human step labels, rollout labels, LLM judges, or pretrained process evaluators.
- We instantiate marginal-contribution supervision through Step Value Integration, which attributes a learned differentiable trajectory value to individual reasoning steps under a temporally constrained cooperative-game view.
- We add two stabilizing supervision signals, model-counterfactual consistency and cross-trajectory inherent scoring, and validate the resulting supervision through controlled same-backbone comparisons, process evaluation, Best-of-N reranking, downstream reinforcement learning, and experiments in verifiable code generation.

2 Related Work

Process Reward Model Process reward models (PRMs) provide fine-grained reward signals at each reasoning step and have substantially improved LLM performance on complex multi-step tasks such as mathematical problem solving and code generation [34, 20]. Compared with outcome reward models that supervise only the final answer, PRMs can localize intermediate errors more precisely, improve temporal credit assignment, and mitigate the alignment gap where the answer is correct but the reasoning process is flawed. As a result, PRMs have become a core technique for test-time scaling, such as Best-of-N [21], and reinforcement learning [19, 38, 7]. Most PRMs are trained to judge whether an intermediate step is locally correct, but correctness alone does not specify how much credit the step deserves for the final outcome. SPRM complements this view by introducing a fairness-oriented credit criterion: it measures step credit through marginal contribution, so that process rewards reflect how each reasoning step changes the trajectory outcome rather than only whether the step appears plausible in isolation.

Automated Annotation and Credit Redistribution Automated process-labeling methods include MC-based step annotation [36, 44, 45, 6, 11] and LLM-as-a-judge labeling [42, 14]. MC labels require large rollout budgets and can be noisy, while judge labels can inherit model biases and prompt sensitivity. Hybrid pipelines combine the two to balance quality and scalability [37, 45], but still require extra rollouts or judge calls. Credit-assignment methods for delayed rewards provide a complementary view: RUDDER [1] redistributes terminal returns through contribution analysis while preserving the trajectory objective in a return-equivalent decision process. Shapley-based model explanations offer a principled allocation rule, but their outputs depend on how the value function, background distribution, and explanation context are operationalized [32]. SPRM instead learns an internal differentiable value function and assigns Aumann-Shapley-style credit from outcome-only trajectories, avoiding external supervision or attribution sources such as rollouts, LLM judges, pretrained evaluators, or background explanation distributions.

3 Learning SPRM from Outcome-Only Trajectories

3.1 Problem Formulation and Motivation

An outcome-supervised reasoning example consists of a question or instruction q_i, a sequence of intermediate reasoning steps $s_{i,1}, \ldots, s_{i,T_i}$, and a terminal ground-truth reward $r_i \in \{0, 1\}$. We write the resulting trajectory as $\tau_i = (q_i, s_{i,1}, \ldots, s_{i,T_i})$ and the dataset as $\mathcal{D} = \{(\tau_i, r_i)\}_{i=1}^{N}$.

In an ordered reasoning trajectory, a step's apparent utility is entangled with its position: later steps can benefit from a correct prefix, while earlier steps may be under-credited once later steps complete the solution. We therefore treat outcome-only supervision as a fair credit-allocation problem, where the trajectory reward is produced by cooperation among steps and should be assigned by marginal contribution rather than temporal artifacts.

Following Shapley et al. [30], we define a finite cooperative game as a pair (N, v), where N is the participant set and $v : 2^N \to \mathbb{R}$ gives the payoff $v(S)$ of each coalition $S \subseteq N$, with $v(\emptyset) = 0$. For a fixed participant set N, an allocation rule ψ maps a value function v to a credit vector $\psi(v) = (\phi_j(v))_{j \in N}$, where $\phi_j(v)$ is the credit assigned to participant j. We call an allocation fair if it satisfies the four Shapley axioms in Table 1.

Table 1: Classical Shapley axioms for a finite cooperative game.

Axiom	Meaning
Efficiency	The total payoff is fully allocated: $\sum_{j \in N} \phi_j(v) = v(N) - v(\emptyset)$.
Symmetry	If two participants make the same marginal contribution to every coalition, they receive the same credit.
Dummy participant	If a participant never changes any coalition value, its credit is zero.
Additivity / Linearity	For two games v and w, the allocation for $av + bw$ is $a\phi(v) + b\phi(w)$.

To connect this game-theoretic formulation to reasoning, we map the reasoning steps in trajectory τ_i to the participant set $\mathcal{N}_i = \{s_{i,1}, \ldots, s_{i,T_i}\}$, and use the trajectory outcome as the shared payoff. For a coalition $S \subseteq \mathcal{N}_i$ with value $v(S)$, the classical Shapley credit of step $s_{i,t}$ is

$$\phi_{i,t}^{\mathrm{sha}} := \frac{1}{T_i!} \sum_{\pi \in \Pi(\mathcal{N}_i)} \Big[v\big(\mathrm{Pre}_\pi(s_{i,t}) \cup \{s_{i,t}\}\big) - v\big(\mathrm{Pre}_\pi(s_{i,t})\big) \Big], \tag{1}$$

where $\Pi(\mathcal{N}_i)$ is the set of permutations and $\mathrm{Pre}_\pi(s_{i,t})$ denotes predecessors of $s_{i,t}$ under π.

Shapley allocation is appealing because, in unordered cooperative games, it provides a fair marginal-contribution allocation. Reasoning trajectories, however, are not exchangeable coalitions: their steps must respect temporal-causal dependencies, so the classical symmetry axiom cannot be imposed literally. We call this restricted setting a *temporally constrained reasoning game*, which assumes the temporal-order condition in Appendix A.2, and define its *admissible-order Shapley-style credit* as the Shapley marginal-contribution average taken only over temporally admissible orders; see Appendix A.3 for the resulting gap from classical Shapley symmetry. Let $P_{i,t} = \{s_{i,1}, \ldots, s_{i,t}\}$ and $P_{i,0} = \emptyset$.

Lemma 3.1. *In a temporally constrained reasoning game where the only admissible order is $1 \prec 2 \prec \cdots \prec T_i$, the admissible-order Shapley-style credit of step $s_{i,t}$ is the prefix value difference*

$$\phi_{i,t}^{\mathrm{ord}} = v(P_{i,t}) - v(P_{i,t-1}). \tag{2}$$

Appendix A.4 provides the proof. The lemma shows that this ordered construction no longer averages over multiple orders, but collapses to prefix differences. This form is operationally unattractive: it requires accurate multi-step value regression over prefixes, and any prediction error can propagate along the reasoning chain. We therefore move from discrete prefix differences to a continuous allocation view based on Aumann-Shapley values [2], which generalize Shapley's average over permutations to the continuous setting for measuring marginal contribution. Integrated Gradients provides a concrete instance of this idea and is the unique continuous attribution rule satisfying Shapley-aligned axioms [33, 32]. Building on this principle, we propose **Step Value Integration** (SVI): a lightweight outcome-trained trajectory value function whose gradients are integrated to assign step credit.

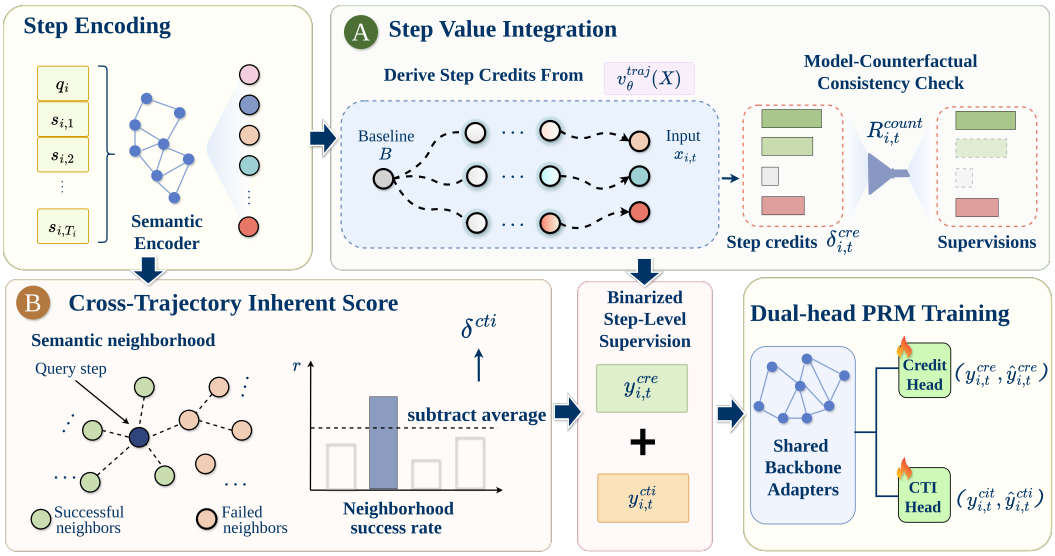


Figure 2: Overview of SPRM. From outcome-only trajectories, SPRM encodes questions and steps, derives complementary SVI and CTI supervision signals, binarizes them to train a dual-head PRM, and combines both heads at inference time for step rewards.

3.2 Step Value Integration with Model-Counterfactual Consistency

For each trajectory $\tau_i = (q_i, s_{i,1}, \ldots, s_{i,T_i})$, we use an encoder with sufficient semantic encoding capacity, such as an embedding model or an LLM, to encode the question and steps as $X_i = \text{encoder}(\tau_i) = [x_{i,0}, x_{i,1}, \ldots, x_{i,T_i}] \in \mathbb{R}^{(T_i+1)\times d}$, where $x_{i,0}$ is the question representation and $x_{i,t}$ represents step $s_{i,t}$, so X_i can be viewed as the participants $\mathcal{N}_i$ in this game. Note that we also encode the question itself; we refer to its representation $x_{i,0}$ as *question_end* and discuss its role in Table 4. We train a lightweight differentiable trajectory value function $v_\theta^{\text{traj}} : \mathbb{R}^{(T_i+1)\times d} \to [0,1]$ from final rewards only:

$$\theta^\star := \arg\min_\theta \mathbb{E}_{(\tau,r)\sim\mathcal{D}} \left[\text{BCE}\Big(r, v_\theta^{\text{traj}}(X)\Big) \right]. \tag{3}$$

The trajectory value function converts step-by-step value estimation into a global value prediction problem, making the target learnable by a lightweight model and removing the need for a strong external value model. SVI assigns each step a share of the change in predicted trajectory value from the baseline to the observed trajectory. Let $b \in \mathbb{R}^d$ be a semantically empty baseline, let $B_i = [b, \ldots, b]$, and define the straight-line path function $\gamma_i(\alpha) = B_i + \alpha(X_i - B_i)$. The Aumann-Shapley-style credit of step $s_{i,t}$ is

$$\delta_{i,t}^{\text{cre}} := (x_{i,t} - b)^\top \int_0^1 \nabla_{x_{i,t}} v_{\theta^\star}^{\text{traj}}(\gamma_i(\alpha))\, d\alpha. \tag{4}$$

The straight-line path specifies how credit is allocated; unlike a causal intervention, it does not require intermediate states to correspond to valid textual reasoning. This attribution is interpreted under the assumptions in Appendix A.2. We analyze the advantages of this SVI credit over the naive Shapley delta-value form in Table 4.

Lemma 3.2. *For a fixed trajectory value function and baseline, with the straight-line path function γ_i, the SVI credits in Equation* (4) *satisfy Shapley-like properties.*

The proof is provided in Appendix A.4, and the Shapley-like properties are discussed in Appendix A.3. In nonlinear value models, SVI distributes interaction credit across steps and can be affected by local sensitivity and numerical integration noise. We therefore introduce model-counterfactual consistency, a representation-ablation check within the learned trajectory value model. It combines the direct removal response, which reflects interactions involving the current step, with pairwise interaction evidence from preceding steps in a local window to filter SVI labels by sign agreement. Let $X_i^{(-t)}$

replace $x_{i,t}$ with b, and let $X_i^{(-t,-u)}$ replace both $x_{i,t}$ and $x_{i,u}$.

$$R_{i,t}^{\mathrm{dir}} = v_{\theta^\star}^{\mathrm{traj}}(X_i) - v_{\theta^\star}^{\mathrm{traj}}(X_i^{(-t)}), \quad R_{i,(t,u)}^{\mathrm{int}} = R_{i,t}^{\mathrm{dir}} - v_{\theta^\star}^{\mathrm{traj}}(X_i^{(-u)}) + v_{\theta^\star}^{\mathrm{traj}}(X_i^{(-t,-u)}). \tag{5}$$

$R_{i,t}^{\mathrm{dir}}$ measures the predicted value difference before and after ablating step t's representation, whereas $R_{i,(t,u)}^{\mathrm{int}}$ measures the pairwise interaction between step representations t and u. We define a local window $\mathcal{W}(t) = \{u \mid 1 \le u < t,\ t-u \le w\}$ to restrict the view of each step to nearby previous steps, and combine the interaction terms with $R_{i,t}^{\mathrm{dir}}$ to obtain the final consistency score:

$$R_{i,t}^{\mathrm{count}} := R_{i,t}^{\mathrm{dir}} + \beta \frac{\sum_{u \in \mathcal{W}(t)} \mathbf{1}\Big(R_{i,(t,u)}^{\mathrm{int}} > 0\Big)\ R_{i,(t,u)}^{\mathrm{int}}}{\sum_{u \in \mathcal{W}(t)} \mathbf{1}\Big(R_{i,(t,u)}^{\mathrm{int}} > 0\Big) + \epsilon}. \tag{6}$$

We omit negative pairwise interactions because they may repeatedly penalize later steps for the same upstream error, even when those steps are locally coherent. The direct term $R_{i,t}^{\mathrm{dir}}$ still retains negative evidence from representation ablation. As a conservative consistency filter, we retain an SVI label only when $\delta_{i,t}^{\mathrm{cre}}$ and $R_{i,t}^{\mathrm{count}}$ share the same nonzero sign, allowing both positive and negative labels.

3.3 Cross-Trajectory Inherent (CTI) Score

In temporal-causal reasoning, each step is interpreted under its current context, so its credit can be strongly affected by the surrounding trajectory. For example, a useful step may be underestimated when it appears after an incorrect prefix. To reduce this variance, CTI estimates the inherent latent value of a step by comparing semantically similar steps across trajectories and measuring their association with final success. We use this inherent score as auxiliary supervision for more stable PRM training. For step representation $x_{i,t}$, retrieve top-M neighbors from other trajectories and denote them by $\mathcal{H}_M(i,t)$. With similarity weights $\omega_{i,t,j,\ell} \ge 0$ and $\epsilon_\omega > 0$, define

$$\hat{r}_{i,t} := \frac{\sum_{(j,\ell) \in \mathcal{H}_M(i,t)} \omega_{i,t,j,\ell} r_j}{\sum_{(j,\ell) \in \mathcal{H}_M(i,t)} \omega_{i,t,j,\ell} + \epsilon_\omega}. \tag{7}$$

Comparing this neighborhood success rate with the dataset average $\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i$ gives the cross-trajectory inherent score

$$\delta_{i,t}^{\mathrm{cti}} := \hat{r}_{i,t} - \bar{r}. \tag{8}$$

Positive $\delta_{i,t}^{\mathrm{cti}}$ means that the step pattern appears in more successful neighborhoods than average. This cross-trajectory inherent score reshapes the final step reward away from noisy single-trajectory evidence.

3.4 PRM Training and Inference

The PRM uses a shared backbone with step-credit and cross-trajectory heads, parameterized by θ_{PRM}. For step $s_{i,t}$, the heads predict $\hat{y}_{i,t}^{\mathrm{cre}} \in (0,1)$ and $\hat{y}_{i,t}^{\mathrm{cti}} \in (0,1)$. We binarize the constructed signals as $y_{i,t}^{\mathrm{cre}} = \mathbf{1}\big(\delta_{i,t}^{\mathrm{cre}} > 0\big)$ and $y_{i,t}^{\mathrm{cti}} = \mathbf{1}\big(\delta_{i,t}^{\mathrm{cti}} > 0\big)$. The step-credit label is used only when the attribution sign agrees with the model-counterfactual consistency sign:

$$m_{i,t} := \mathbf{1}\big(\delta_{i,t}^{\mathrm{cre}} \cdot R_{i,t}^{\mathrm{count}} > 0\big). \tag{9}$$

The final objective is

$$\theta_{\mathrm{PRM}}^\star := \arg\min_{\theta_{\mathrm{PRM}}} \mathbb{E}_{(\tau_i, r_i) \sim \mathcal{D}} \left[\sum_{t=1}^{T_i} m_{i,t}\, \mathrm{BCE}\big(y_{i,t}^{\mathrm{cre}}, \hat{y}_{i,t}^{\mathrm{cre}}\big) + \sum_{t=1}^{T_i} \mathrm{BCE}\big(y_{i,t}^{\mathrm{cti}}, \hat{y}_{i,t}^{\mathrm{cti}}\big) \right]. \tag{10}$$

At inference time, the final step reward is

$$\hat{y}_{i,t} := \lambda \hat{y}_{i,t}^{\mathrm{cre}} + (1-\lambda)\hat{y}_{i,t}^{\mathrm{cti}}, \tag{11}$$

where $\lambda \in [0,1]$ controls the trade-off between trajectory-internal step credit and cross-trajectory inherent scoring.

Table 2: ProcessBench results. $\Delta = \text{SPRM} - \text{baseline}$ for the average F1; positive values indicate higher scores for SPRM. Data for models marked with an asterisk ($*$) are sourced from [45].

Model	GSM8K			MATH			OLYMPIADBENCH			OMNIMATH			Avg_F1	Δ
	Err	Cor	F1	Err	Cor	F1	Err	Cor	F1	Err	Cor	F1		
Closed-source judges														
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	-1.9
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	-27.9
Open-source judges														
Llama-3.3-70B-Instruct*	72.5	96.9	82.9	43.3	83.2	59.4	31.0	94.1	46.7	28.2	90.5	43.0	58.0	**+2.0**
Qwen2.5-Math-72B-Instruct*	49.8	96.9	65.8	36.0	94.3	52.1	19.5	97.3	32.5	19.0	96.3	31.7	45.5	**+14.5**
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	-1.2
PRMs														
Math-Shepherd-7B	33.1	88.7	48.2	17.8	84.1	29.4	15.0	68.2	24.6	15.2	74.2	25.2	31.9	**+28.1**
RLHFlow-Mistral-8B	34.2	98.4	50.9	21.9	72.4	33.6	8.6	47.8	14.6	9.6	45.6	15.9	28.8	**+31.2**
RLHFlow-DeepSeek-8B	24.6	98.4	39.4	22.6	78.1	35.0	10.3	51.3	17.1	10.3	51.9	17.2	27.2	**+32.8**
Skywork-PRM-7B	58.4	70.0	63.7	40.7	62.3	49.3	16.0	20.0	17.8	12.3	34.4	18.2	37.3	**+22.7**
ReasonEval-7B	19.8	97.4	32.9	27.4	82.5	41.2	22.7	64.6	33.6	21.5	72.6	33.1	35.2	**+24.8**
Qwen2.5-Math-7B-Math-Shepherd*	46.4	95.9	62.4	18.9	96.6	31.6	7.4	93.8	13.7	4.0	95.6	7.7	28.9	**+31.1**
Qwen2.5-Math-7B-PRM800K	52.8	94.7	67.8	47.8	91.2	62.7	33.7	85.2	48.3	27.1	87.2	41.3	55.0	**+5.0**
OVM-Qwen3-4B	4.8	56.5	8.9	37.7	85.5	52.3	31.0	79.9	44.7	23.3	80.5	36.2	35.5	**+24.5**
PQM-DeepSeek-zeta4-7B	31.9	40.9	35.8	22.2	19.7	20.9	8.9	6.2	7.3	11.1	15.8	13.0	19.2	**+40.8**
SPRM-Qwen3-4B	55.5	94.8	70.1	52.4	76.0	62.8	48.1	60.0	53.2	48.6	60.2	53.8	**60.0**	-

Table 3: PRMBench results. $\Delta = \text{SPRM} - \text{baseline}$ for the Overall score; positive values indicate higher scores for SPRM. Data for models marked with an asterisk ($*$) are sourced from [31].

Model	Simplicity			Soundness					Sensitivity				Overall	Δ
	NR	NCL	Avg	ES	SC	DC	CI	Avg	PS	DR	MS	Avg		
Closed-source judges														
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	-4.4
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	-6.4
PRMs														
Math-Shepherd-7B	42.3	47.3	44.8	50.1	42.3	41.2	48.0	45.4	43.1	49.6	90.1	60.9	46.6	**+15.8**
RLHFLOW-Mistral-8B	46.3	47.1	46.7	56.5	54.8	54.0	63.9	57.3	51.0	55.9	98.2	68.4	54.2	**+8.2**
RLHFLOW-DeepSeek-8B	46.4	48.5	47.5	55.0	54.5	53.1	65.6	57.1	48.8	55.0	99.7	67.8	53.8	**+8.6**
Skywork-PRM-7B	40.5	48.1	44.3	45.5	36.8	38.0	43.3	40.9	42.7	45.5	93.3	60.5	43.3	**+19.1**
ReasonEval-7B	49.2	57.1	53.2	62.1	66.0	61.5	65.8	63.9	55.6	58.0	99.1	70.9	59.8	**+2.6**
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	59.1	**+3.3**
OVM-Qwen3-4B	50.0	43.4	46.7	45.5	48.7	46.4	47.3	47.0	46.3	47.7	95.2	63.1	47.2	**+15.2**
PQM-DeepSeek-zeta4-7B	17.7	25.7	21.7	22.9	16.1	16.6	18.4	18.5	21.6	20.8	43.7	28.7	20.3	**+42.1**
SPRM-Qwen3-4B	52.5	64.9	58.7	67.5	59.8	58.6	65.1	62.8	60.0	63.9	99.1	74.3	**62.4**	-

4 Experiments

4.1 Experimental Settings

Evaluation Datasets and Metrics Our experimental design evaluates SPRM along three complementary dimensions: **first-error localization**, which tests whether a PRM can locate the first wrong step in an erroneous chain while recognizing all-correct chains; **credit assignment**, which probes whether reward reflects marginal contribution rather than redundant or logically flawed steps; and **reasoning-time selection**, which tests whether process rewards select better complete reasoning trajectories during Best-of-N inference [20]. We instantiate these dimensions with ProcessBench [46], PRMBench [31], and Best-of-N reranking, respectively. For the latter, we rerank trajectories from AceMath-RewardBench [23] and report final-answer accuracy across problem difficulties and search ranges, averaged over 10 random seeds. We visualize four representative datasets in the main text, aggregate step scores according to each model's originally recommended configuration, and defer metric definitions and complete dataset-wise results to Appendices B.2 and B.5.

Baselines We select representative baselines by supervision source, as summarized in Table 7: human-labeled PRMs (Qwen2.5-Math-7B-PRM800K [45] and ReasonEval-7B [39]), Monte Carlo-labeled PRMs (Math-Shepherd-7B [36] and RLHFlow [12]), outcome-reward broadcasting (OVM-Qwen3-4B [41]), first-error boundary modeling (PQM-DeepSeek-zeta4-7B [19]), and curated process supervision (Skywork-PRM-7B [15]). SPRM is trained from Math-Shepherd trajectories [36] without using its intermediate step rewards.

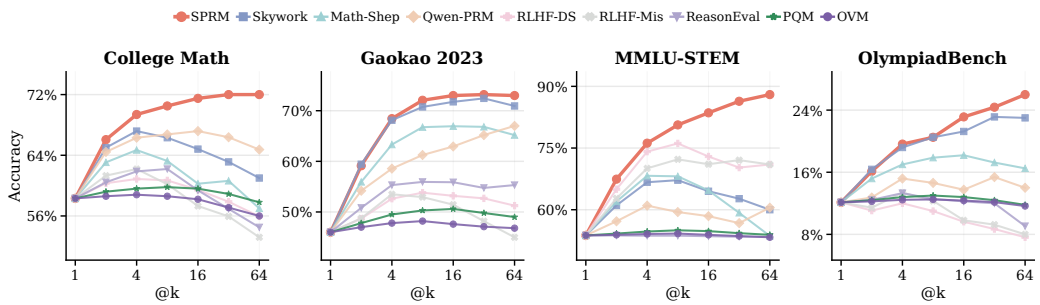


Figure 3: Best-of-N inference with different PRMs on four representative datasets. Each subplot reports accuracy as a function of k on one dataset.

4.2 Main Results

SPRM Addresses the Three Targeted Evaluation Properties **First-error localization.** On ProcessBench, SPRM achieves the best average F1 of 60.0, improving over Math-Shepherd-7B and RLHFlow by 28.1–32.8 points and over the same-backbone OVM-Qwen3-4B by 24.5 points. This gap suggests that rollout labels and outcome broadcasting still blur the marginal cause of failure, while SPRM better separates the first harmful step from its surrounding context without simply accepting more trajectories as correct. **Marginal-contribution preferences.** On PRMBench, SPRM obtains the best overall score among open PRMs at 62.4, with strong *Simplicity* and *Sensitivity* scores of 58.7 and 74.3; its NCL score of 64.9 and PS score of 60.0 show that it better down-weights redundant reasoning and penalizes harmful perturbations. **Reasoning-time selection.** Figure 3 shows that SPRM scales reliably as k increases to 64, reaching 72.0 on CollegeMath, 73.0 on Gaokao2023en, 97.0 on GSM8K, and 90.0 on MATH500. By contrast, OVM drops on CollegeMath and MATH500 as k grows from 2 to 64, while Math-Shepherd largely saturates, indicating that marginal-contribution supervision gives a more stable selection signal under larger search pools.

5 Analysis

This section asks four **R**esearch **Q**uestions (**RQ**) about the mechanisms and cross-domain applicability of the constructed supervision. RQ1–RQ3 isolate the roles of SPRM's supervision components while keeping the backbone, training distribution, and optimization setup fixed, so performance differences primarily reflect the supervision signal itself. These three experiments use ProcessBench and report *Err*, *Cor*, and F1, as summarized in Table 4. RQ4 tests transfer of the supervision framework to code generation through Best-of-N selection accuracy. The four RQs are:

RQ1. *Can SVI provide more learnable step credit than temporal value differences?*

RQ2. *Why choose R^{count}-consistency instead of LLM-consistency?*

RQ3. *Can cross-trajectory inherent score reduce context-induced credit fluctuations?*

RQ4. *Can SPRM's outcome-to-step supervision transfer beyond mathematical reasoning?*

RQ1: Can SVI provide more learnable step credit than temporal value differences? To answer RQ1, we construct three PRM training signals in Table 4: delta-value labels, SVI labels without *question_end*, and full SVI labels with *question_end*. This ablation is designed to test whether the supervision quality improves when step credit is derived from one trajectory-level predictor, and whether the reasoning-start boundary should participate in the allocation.

The motivation is that delta-value labels propagate prefix-value errors. If $\hat{v}(P_t) = v^\star(P_t) + \epsilon_t$, then the sign of the delta label is stable only when $|\epsilon_t - \epsilon_{t-1}| < |v^\star(P_t) - v^\star(P_{t-1})|$; small prefix errors can therefore flip labels when the true marginal gap is small. SVI does not remove value-estimation error, but changes it from a difference of two prefix residuals into the path-integrated attribution of one trajectory residual. Let $v_\theta^{\text{traj}} = v^{\star,\text{traj}} + e$. By completeness [33], the residual in $v_\theta^{\text{traj}}(X) - v_\theta^{\text{traj}}(B)$ is assigned over the encoded participants used in the allocation. This explains why full SVI improves

Table 4: ProcessBench ablations isolating SPRM's three design modules: SVI step-credit source, model-counterfactual consistency mechanism, cross-trajectory inherent score, and the SVI/CTI fusion weight λ, to assess their individual contributions. Gray shading marks the configurations used in our method.

Component	Setting	Err	Cor	F1	Component	Setting	Err	Cor	F1
Step-credit source	Delta value	24.2	75.6	35.5	*CTI score*	SVI only	42.7	**79.0**	55.4
	SVI w/o *question_end*	31.4	42.6	36.1		SVI + random	43.8	74.6	55.2
	SVI	**37.7**	**76.5**	**50.2**		CTI only	26.0	46.9	33.4
Consistency Mechanism	No consistency	37.7	76.5	50.2		SVI + CTI	**51.2**	73.3	**60.0**
	LLM-consistency	**47.3**	67.0	**55.4**	$\lambda, (1-\lambda)$	$(1.00, 0.00)$	42.7	79.0	55.4
						$(0.75, 0.25)$	40.9	**83.1**	54.7
						$(0.50, 0.50)$	**51.2**	73.3	**60.0**
						$(0.25, 0.75)$	34.3	60.6	43.8
	R^{count}-consistency	42.7	**79.0**	**55.4**		$(0.00, 1.00)$	26.0	46.9	33.4

Table 5: Cross-domain transfer to code generation: selection accuracy (%) on LiveCodeBench. Entries report mean $\pm$ standard deviation over three candidate-sampling seeds; BoN@32 uses the entire fixed pool and thus has zero standard deviation. Base is model accuracy before training; Pass@K is the upper bound indicating whether any candidate passes all tests. Boldface marks the best mean excluding this upper bound.

Method	BoN@4	BoN@8	BoN@16	BoN@32
Pass@K (upper bound)	49.1 ± 0.3	52.3 ± 0.1	53.0 ± 0.6	57.2 ± 0.0
Base (before training)	40.0 ± 0.5	40.9 ± 0.7	41.7 ± 0.2	46.8 ± 0.0
OVM	43.5 ± 0.3	44.0 ± 0.8	43.8 ± 1.0	48.6 ± 0.0
SPRM	$\mathbf{43.7 \pm 0.8}$	$\mathbf{46.2 \pm 0.7}$	$\mathbf{47.9 \pm 1.1}$	$\mathbf{52.1 \pm 0.0}$

average F1 from 35.5 to 50.2. It also explains the boundary ablation: the reasoning start may itself carry value mismatch, and excluding *question_end* forces this error onto the first reasoning step, reducing F1 to 36.1. Thus, RQ1 is answered positively: SVI gives more learnable step supervision by avoiding prefix-error differencing, provided that the allocation includes the reasoning-start participant.

RQ2: Why choose R^{count}-consistency instead of LLM-consistency? To answer RQ2, we design a consistency ablation that trains PRMs with three filters on SVI credit: no consistency filter, an LLM-consistency filter using DeepSeek-V3.2 [8] (2026-03-24), and our model-counterfactual consistency filter, denoted by R^{count}. This experiment directly tests whether an external strong LLM judge is a better consistency source than the trajectory-value-based interaction score. LLM-consistency gives the highest *Err* score, 47.3, indicating stricter error correction, which is expected from a strong model with substantial logical checking ability. However, it lowers *Cor* to 67.0, consistent with observations that LLMs can be unstable when producing long structured arrays [16]. By contrast, R^{count}-consistency preserves the strongest *Cor* score of 79.0 while matching the best F1 of 55.4 and improving over the no-consistency F1 of 50.2. This follows from its design: $R^{\text{int}}_{i,(t,u)}$ in Equation (5) measures whether a step supports local interactions with previous steps, and Equation (6) only aggregates positive interaction terms. Thus, R^{count} emphasizes steps that participate in a fluent reasoning chain rather than aggressively penalizing every local imperfection, which helps preserve truly useful steps for PRM learning.

RQ3: Can cross-trajectory inherent score reduce context-induced credit fluctuations? To answer RQ3, we analyze whether cross-trajectory evidence can stabilize trajectory-specific credit. SVI is trajectory-conditioned: the same step pattern may receive different credit when it appears after a flawed prefix, within an easy problem, or near a decisive later correction. CTI addresses this variance by estimating a non-parametric conditional-success tendency: $\delta^{\text{cti}}(x) \approx P(\text{success} \mid \text{similar step appears}) - P(\text{success})$. SVI asks how the current step contributes within its current trajectory, whereas CTI asks whether semantically similar steps are statistically associated with successful trajectories in the data distribution.

We therefore design the CTI ablations to separate true cross-trajectory complementarity from generic smoothing: "SVI only" uses trajectory-internal credit, "CTI only" uses the inherent score alone, "SVI

+ CTI" combines the two, and "SVI + random" replaces semantic neighbors with random ones. "CTI only" is weak because it lacks the current trajectory's causal context, but adding CTI to SVI improves average F1 from 55.4 to 60.0. Replacing retrieved neighbors with random ones drops F1 to 55.2, showing that the gain comes from semantic cross-trajectory evidence rather than arbitrary averaging. The head-mixing rows give the same conclusion: too much CTI weight removes trajectory-internal context, while the balanced setting gives the best F1. Together with the trajectory-mean δ^{cti} AUC of 0.856 against final reward in Appendix B.4.4, these results support CTI as a stabilizer for SVI rather than a replacement for trajectory-specific credit.

RQ4: Can SPRM's outcome-to-step supervision transfer beyond mathematical reasoning? To answer RQ4, we examine whether outcome-based step credit also benefits code generation by retraining SPRM on code trajectories with the same PRM training configuration as in the mathematics experiments. Qwen3-4B-Instruct-2507 generates the BigCodeBench [47] training trajectories, segmented into function-level steps following FunPRM [43]. The dataset contains 4,697 trajectories and 15,040 steps; only programs passing all unit tests receive positive terminal labels, while partially passing and failing programs receive negative labels. We train a same-backbone OVM on the same data by broadcasting terminal labels to all steps. This comparison tests whether allocating outcome feedback to individual steps is more effective than assigning the same terminal label to every step. For evaluation, the same generator produces a fixed pool of 32 candidates per problem on the LiveCodeBench v6 [17] slice starting on 2025-02-01, retaining 134 executable problems. We sample and rerank $K \in \{4, 8, 16, 32\}$ candidates per problem and report mean accuracy and standard deviation over three candidate-sampling seeds.

Table 5 shows that SPRM has higher mean accuracy than OVM at all four budgets: the gap is small at BoN@4, and increases to 2.2, 4.1, and 3.5 percentage points at BoN@8, BoN@16, and BoN@32, respectively. At BoN@32, SPRM reaches 52.1% compared with OVM's 48.6%. The gains over terminal-label broadcasting suggest that SPRM can extract useful step-level supervision in code generation, providing evidence of its transfer potential beyond mathematical reasoning under reliable terminal verification. Extending this framework to open-ended tasks or noisier reward settings is a natural direction for future work.

6 Conclusion

We presented SPRM, an outcome-only framework that derives process reward supervision from sparse terminal feedback via an ordered cooperative game. By combining Aumann-Shapley-style attribution, model-counterfactual consistency, and cross-trajectory inherent scoring, SPRM produces scalable step-level supervision without strong external supervision. Experiments on verifiable reasoning tasks show consistent gains in first-error localization, trajectory-level evaluation, and Best-of-N reranking. A key limitation is that the current method assumes verifiable outcome rewards, and CTI depends on embedding quality for reliable semantic retrieval. Future work will extend this perspective to reward modeling for tasks where answers cannot be reliably verified.

Acknowledgments

This work is supported by the National Key R&D Program of China (2026YFE0199800), Science and Technology Development Fund of Macao S.A.R (FDCT) under 0123-2022-AFJ, 0081-2022-A2, and 0074-2025-AMJ, National Natural Science Foundation of China (62376263), Natural Science Foundation of Guangdong (2024A1515030209), and Shenzhen Science and Technology Innovation Commission (JCYJ20230807140507015).

References

[1] Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: return decomposition for delayed rewards. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13544–

13555, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/16105fb9cc614fc29e1bda00dab60d41-Abstract.html.

[2] Robert J Aumann and Lloyd S Shapley. *Values of non-atomic games*. Princeton University Press, 2015.

[3] Chuxue Cao, Mengze Li, Juntao Dai, Jinluan Yang, Zijian Zhao, Shengyu Zhang, Weijie Shi, Chengzhong Liu, Sirui Han, and Yike Guo. Towards advanced mathematical reasoning for llms via first-order logic theorem proving. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 12429–12449. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.628. URL https://doi.org/10.18653/v1/2025.emnlp-main.628.

[4] Qi Cao, Ruiyi Wang, Ruiyi Zhang, Sai Ashish Somayajula, and Pengtao Xie. Dreamprm: Domain-reweighted process reward model for multimodal reasoning. *CoRR*, abs/2505.20241, 2025. doi: 10.48550/ARXIV.2505.20241. URL https://doi.org/10.48550/arXiv.2505.20241.

[5] Wen-Tse Chen, Jiayu Chen, Fahim Tajwar, Hao Zhu, Xintong Duan, Ruslan Salakhutdinov, and Jeff G. Schneider. Retrospective in-context learning for temporal credit assignment with large language models. *CoRR*, abs/2602.17497, 2026. doi: 10.48550/ARXIV.2602.17497. URL https://doi.org/10.48550/arXiv.2602.17497.

[6] Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Nijun Li, Tao Gui, Yun Li, Qi Zhang, and Xuanjing Huang. Better process supervision with bi-directional rewarding signals. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14471–14485, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.747. URL https://aclanthology.org/2025.findings-acl.747/.

[7] Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *CoRR*, abs/2502.10325, 2025. doi: 10.48550/ARXIV.2502.10325. URL https://doi.org/10.48550/arXiv.2502.10325.

[8] DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025. doi: 10.48550/ARXIV.2512.02556. URL https://doi.org/10.48550/arXiv.2512.02556.

[9] Jasper Dekoninck, Ivo Petrov, Kristian Minchev, Mislav Balunovic, Martin T. Vechev, Miroslav Marinov, Maria Drencheva, Lyuba Konova, Milen Shumanov, Kaloyan Tsvetkov, Nikolay Drenchev, Lazar Todorov, Kalina Nikolova, Nikolay Georgiev, Vanesa Kalinkova, and Margulan Ismoldayev. The open proof corpus: A large-scale study of llm-generated mathematical proofs. *CoRR*, abs/2506.21621, 2025. doi: 10.48550/ARXIV.2506.21621. URL https://doi.org/10.48550/arXiv.2506.21621.

[10] Leander Diaz-Bone, Marco Bagatella, Jonas Hübotter, and Andreas Krause. DISCOVER: automated curricula for sparse-reward reinforcement learning. *CoRR*, abs/2505.19850, 2025. doi: 10.48550/ARXIV.2505.19850. URL https://doi.org/10.48550/arXiv.2505.19850.

[11] Yuyang Ding, Xinyu Shi, Juntao Li, Xiaobo Liang, Zhaopeng Tu, and Min Zhang. SCAN: self-denoising monte carlo annotation for robust process reward learning. *CoRR*, abs/2509.16548, 2025. doi: 10.48550/ARXIV.2509.16548. URL https://doi.org/10.48550/arXiv.2509.16548.

[12] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=a13aYUU9eU.

[13] Kefan Dong and Tengyu Ma. STP: self-play LLM theorem provers with iterative conjecturing and proving. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/dong25h.html.

[14] Keyu Duan, Zichen Liu, Xin Mao, Tianyu Pang, Changyu Chen, Qiguang Chen, Michael Qizhe Shieh, and Longxu Dou. Efficient process reward model training via active learning. *CoRR*, abs/2504.10559, 2025. doi: 10.48550/ARXIV.2504.10559. URL https://doi.org/10.48550/arXiv.2504.10559.

[15] Jujie He, Tianwen Wei, Rui Yan, Jiacai 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. Skywork-o1 open series, November 2024. URL https://doi.org/10.5281/zenodo.16998085.

[16] Chashi Mahiul Islam, Alan Villarreal, Mao Nishino, Shaeke Salman, and Xiuwen Liu. Numerical instability and chaos: Quantifying the unpredictability of large language models, 2026. URL https://arxiv.org/abs/2604.13206.

[17] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

[18] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=FPVCb0WMuN.

[19] Wendi Li and Yixuan Li. Process reward model with q-value rankings. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=wQEdh2cgEk.

[20] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[21] Jianghao Lin, Yuanyuan Shi, Xin Peng, Renjie Ding, Hairui Wang, Yuxuan Peng, Bizhe Bai, Weixi Song, Fengshuo Bai, Huacan Chai, Weinan Zhang, Fei Huang, and Ying Wen. Toolprm: Fine-grained inference scaling of structured outputs for function calling. *CoRR*, abs/2510.14703, 2025. doi: 10.48550/ARXIV.2510.14703. URL https://doi.org/10.48550/arXiv.2510.14703.

[22] Anthony Zhe Liu, Xinhe Wang, Jacob Sansom, Yao Fu, Jongwook Choi, Sungryull Sohn, Jaekyeom Kim, and Honglak Lee. Interactive and expressive code-augmented planning with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 20330–20354. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.994/.

[23] Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acemath: Advancing frontier math reasoning with post-training and reward modeling. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 3993–4015. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.206/.

[24] Zihan Ma, Taolin Zhang, Maosong Cao, Junnan Liu, Wenwei Zhang, Minnan Luo, Songyang Zhang, and Kai Chen. Rethinking verification for LLM code generation: From generation to testing. *CoRR*, abs/2507.06920, 2025. doi: 10.48550/ARXIV.2507.06920. URL https://doi.org/10.48550/arXiv.2507.06920.

[25] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=bNtr6SLgZf.

[26] Ulyana Piterbarg, Lerrel Pinto, and Rob Fergus. Training language models on synthetic edit sequences improves code synthesis. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=AqfUa08PCH.

[27] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[29] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

[30] Lloyd S Shapley et al. A value for n-person games. 1953.

[31] Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 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), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 25299–25346. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1230/.

[32] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pages 9269–9278. PMLR, 2020. URL http://proceedings.mlr.press/v119/sundararajan20b.html.

[33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 3319–3328. PMLR, 2017. URL http://proceedings.mlr.press/v70/sundararajan17a.html.

[34] Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022. doi: 10.48550/ARXIV.2211.14275. URL https://doi.org/10.48550/arXiv.2211.14275.

[35] Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning. *CoRR*, abs/2509.22819, 2025. doi: 10.48550/ARXIV.2509.22819. URL https://doi.org/10.48550/arXiv.2509.22819.

[36] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

[37] Shuai Wang, Zhenhua Liu, Jiaheng Wei, Xuanwu Yin, Dong Li, and Emad Barsoum. Athena: Enhancing multimodal reasoning with data-efficient process reward models. *CoRR*, abs/2506.09532, 2025. doi: 10.48550/ARXIV.2506.09532. URL https://doi.org/10.48550/arXiv.2506.09532.

[38] Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. Agentprm: Process reward models for llm agents via step-wise promise and progress, 2025. URL https://arxiv.org/abs/2511.08325.

[39] Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 27723–27730. AAAI Press, 2025. doi: 10.1609/AAAI.V39I26.34987. URL https://doi.org/10.1609/aaai.v39i26.34987.

[40] Zhaohui Yang, Chenghua He, Xiaowen Shi, Shihong Deng, Linjing Li, Qiyue Yin, and Daxin Jiang. Beyond the first error: Process reward models for reflective mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 4711–4728. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.253/.

[41] Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, volume NAACL 2024 of *Findings of ACL*, pages 858–875. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.55. URL https://doi.org/10.18653/v1/2024.findings-naacl.55.

[42] Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. Versaprm: Multi-domain process reward model via synthetic reasoning data. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zeng25h.html.

[43] Ruiyi Zhang, Peijia Qin, Qi Cao, Eric Xue, and Pengtao Xie. Funprm: Function-as-step process reward model with meta reward correction for code generation. *CoRR*, 2026. URL https://doi.org/10.48550/arXiv.2601.22249.

[44] Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. *CoRR*, abs/2505.14069, 2025. doi: 10.48550/ARXIV.2505.14069. URL https://doi.org/10.48550/arXiv.2505.14069.

[45] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

[46] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 1009–1024. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.50/.

[47] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, and et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

A Assumptions and Theoretical Analysis

Appendix roadmap The appendix is organized into three parts. Appendix A gives the assumptions and theoretical analysis behind the two lemmas used in the main text. It first fixes notation, states the assumptions under which all proofs are interpreted, compares temporal credit assignment with classical Shapley allocation, and then proves the two lemmas used in the main text. Appendix B collects supplementary empirical material, including baseline supervision comparisons, evaluation metrics, compute-cost comparisons, design studies, attribution diagnostics, CTI diagnostics, and complete Best-of-N results. Appendix C gives implementation and environment details for reproducing the reported runs.

A.1 Notation

Table 6: Notation used throughout the appendix, aligned with Section 3.1 and Section 3.2.

Symbol	Meaning
$\mathcal{D}$	Outcome-supervised trajectory dataset.
τ_i	Trajectory containing question q_i and T_i reasoning steps.
r_i	Terminal ground-truth reward of trajectory τ_i.
$\mathcal{N}_i$	Participant set in the discrete reasoning game.
$s_{i,t}$	Step t in trajectory τ_i, treated as a participant in the discrete reasoning game.
$\Pi_{\mathrm{adm}}(\mathcal{N}_i)$	Set of temporally admissible permutations of the participant set $\mathcal{N}_i$.
$\pi^\star$	Unique admissible temporal permutation $(s_{i,1}, s_{i,2}, \ldots, s_{i,T_i})$ under Assumption A1.
$\mathrm{Pre}_\pi(s_{i,t})$	Set of steps that precede $s_{i,t}$ under permutation π.
$P_{i,t}$	Prefix coalition up to step t.
v	Generic value function that maps a coalition, prefix, or encoded trajectory to a scalar value.
$\phi_{i,t}^{\mathrm{ord}}$	Ordered Shapley-style credit assigned to step t under the admissible temporal order.
X_i	Encoded question and step representations.
$x_{i,0}$, $x_{i,t}$	Question representation and step-t representation, respectively.
b, B_i	Semantically empty baseline feature and baseline sequence.
γ_i	Straight-line path function $\gamma_i(\alpha) = B_i + \alpha(X_i - B_i)$.
$\delta_{i,t}^{\mathrm{cre}}$	SVI credit of step t, as defined in Equation (4).
$R_{i,t}^{\mathrm{count}}$	Model-counterfactual consistency score used to mask step-credit labels.
$\delta_{i,t}^{\mathrm{cti}}$	Cross-trajectory inherent score of step t.
ψ	Generic allocation rule for the continuous attribution problem.

A.2 Assumptions

All theoretical statements below use the notation in Table 6 and are interpreted under the following assumptions. **A1. Temporal order.** For each reasoning trajectory, the only admissible causal order is $1 \prec 2 \prec \cdots \prec T_i$; later steps are interpreted after their observed prefixes and are not freely exchangeable with earlier steps. **A2. Fixed attribution problem.** For SVI, all attribution claims are made with respect to a fixed trajectory value function v and a fixed baseline B_i. The path function is the straight-line map $\gamma_i(\alpha) = B_i + \alpha(X_i - B_i)$. **A3. Regularity.** The scalar path function $\alpha \mapsto v(\gamma_i(\alpha))$ is absolutely continuous on $[0, 1]$, and each partial gradient $\nabla_{x_{i,t}} v(\gamma_i(\alpha))$ exists almost everywhere and is integrable along this path. For linearity comparisons, the compared value functions use the same baseline.

A.3 Temporal Shapley Comparison

This section compares the SPRM step-credit term in Equation (4) with classical Shapley axioms. The goal is not to claim that temporal credit assignment exactly matches the classical Shapley value, because reasoning steps are causally ordered and not freely exchangeable. Instead, we clarify which Shapley-inspired properties SPRM satisfies, where the temporal setting departs from the classical

axioms, and what fixed value function, fixed-baseline, and fixed-path constraints are required. The discussion applies only to the attribution signal $\delta_{i,t}^{\mathrm{cre}}$, not to the counterfactual filter, CTI score, or binarized PRM labels.

Classical Shapley axioms For a finite cooperative game with participant set N, value function $v : 2^N \to \mathbb{R}$, and allocation rule ϕ, which maps a value function v to one real-valued credit $\phi(v) = (\phi_j(v))_{j \in N} \in \mathbb{R}^{|N|}$ for each participant, the Shapley value is the unique allocation satisfying the four axioms in Table 1 [30].

These axioms determine a unique notion of marginal-contribution credit: the classical Shapley value averages each participant's marginal contribution over all permutations of the participants. For continuous feature spaces, Sundararajan et al. [33] show that gradient integration is the unique attribution rule satisfying Completeness, Sensitivity, Linearity, and Symmetry-Preserving. These four conditions respectively correspond to the four classical Shapley axioms in Table 1: efficiency, dummy participant, additivity, and symmetry.

Temporal constraints without exchangeability The question in temporally ordered reasoning is whether we can still obtain a Shapley-like credit rule when exchangeability is unavailable, and what is lost relative to the classical characterization. Reasoning trajectories differ from unordered cooperative games because a later step is causally interpreted after its prefix, and permuting steps generally changes the reasoning process. Thus the classical symmetry axiom, and likewise Symmetry-Preserving in the continuous setting, cannot be matched literally. To stay as close as possible to Shapley's fair-allocation principle, we define the trajectory game over the admissible prefix chain $P_{i,0} \subset P_{i,1} \subset \cdots \subset P_{i,T_i}$. On this chain, there is only one admissible temporal order. Thus the Shapley average over permutations collapses to the ordered marginal contribution $v(P_{i,t}) - v(P_{i,t-1})$, as stated in Lemma 3.1. The temporal setting therefore replaces the symmetry-based characterization with three ordered properties that preserve the parts of Shapley most relevant to causal credit assignment: *temporal efficiency*, inherited from efficiency, requires all payoff difference to be allocated over the ordered positions; *prefix dummy*, inherited from the dummy-participant axiom, gives zero credit to a step whose addition to its prefix changes no value; and *linearity*, inherited from additivity, requires allocations to commute with linear combinations of value functions. The gap from classical Shapley is precisely symmetry: we do not assert exchangeability among reasoning steps, but instead use the prefix-order constraint as the admissible causal structure.

Continuous SPRM setting SPRM then moves from discrete prefix values to continuous feature attribution. Under Assumptions A2–A3, Lemma 3.2 shows that the SPRM credit in Equation (4) satisfies the continuous counterparts of efficiency, dummy, and linearity along the path function γ_i. This comparison makes the limitation explicit: SPRM does not satisfy classical Shapley symmetry, because temporal reasoning steps are not exchangeable under the causal order. Thus this appendix positions SPRM as a Shapley-inspired credit assignment rule under temporal-causal constraints, rather than as a full classical Shapley value.

A.4 Lemma Proofs

A.4.1 Proof of the Ordered Prefix-Difference Lemma

Proof of Lemma 3.1. Let $\Pi_{\mathrm{adm}}(\mathcal{N}_i)$ denote the set of temporally admissible permutations of the step set $\mathcal{N}_i = \{s_{i,1}, \ldots, s_{i,T_i}\}$. The admissible-order Shapley-style credit is the marginal-contribution average restricted to this set:

$$\phi_{i,t}^{\mathrm{ord}} := \frac{1}{|\Pi_{\mathrm{adm}}(\mathcal{N}_i)|} \sum_{\pi \in \Pi_{\mathrm{adm}}(\mathcal{N}_i)} \Big[v\big(\mathrm{Pre}_\pi(s_{i,t}) \cup \{s_{i,t}\}\big) - v\big(\mathrm{Pre}_\pi(s_{i,t})\big) \Big]. \tag{12}$$

By Assumption A1, the only admissible order is

$$\pi^\star = (s_{i,1}, s_{i,2}, \ldots, s_{i,T_i}), \qquad \Pi_{\mathrm{adm}}(\mathcal{N}_i) = \{\pi^\star\}. \tag{13}$$

Thus $|\Pi_{\mathrm{adm}}(\mathcal{N}_i)| = 1$, and Equation (12) becomes

$$\phi_{i,t}^{\mathrm{ord}} = v\big(\mathrm{Pre}_{\pi^\star}(s_{i,t}) \cup \{s_{i,t}\}\big) - v\big(\mathrm{Pre}_{\pi^\star}(s_{i,t})\big). \tag{14}$$

Under $\pi^\star$, the predecessors of step $s_{i,t}$ are exactly the previous prefix:

$$\mathrm{Pre}_{\pi^\star}(s_{i,t}) = \{s_{i,1}, \ldots, s_{i,t-1}\} = P_{i,t-1}. \tag{15}$$

Moreover,

$$\mathrm{Pre}_{\pi^\star}(s_{i,t}) \cup \{s_{i,t}\} = \{s_{i,1}, \ldots, s_{i,t}\} = P_{i,t}. \tag{16}$$

Substituting these two identities yields

$$\phi_{i,t}^{\mathrm{ord}} = v(P_{i,t}) - v(P_{i,t-1}), \tag{17}$$

which proves the claim. □

A.4.2 Proof of the SVI Shapley-Like Lemma

Proof of Lemma 3.2. The three Shapley-like properties considered here are discussed and motivated in Appendix A.3. We prove that temporal efficiency, prefix dummy, and linearity survive under the fixed trajectory value function and baseline, with the path function γ_i, in Assumptions A2–A3. The SVI credit is

$$\delta_{i,t}^{\mathrm{cre}} := (x_{i,t} - b)^\top \int_0^1 \nabla_{x_{i,t}} v(\gamma_i(\alpha))\, d\alpha. \tag{18}$$

Temporal efficiency By Assumption A3 and the fundamental theorem of calculus along γ_i,

$$v(X_i) - v(B_i) = \int_0^1 \nabla v(\gamma_i(\alpha))^\top (X_i - B_i)\, d\alpha. \tag{19}$$

Decomposing the inner product over encoded positions gives

$$\sum_{t=0}^{T_i} \delta_{i,t}^{\mathrm{cre}} = v(X_i) - v(B_i). \tag{20}$$

If only reasoning steps are summed, the question representation is excluded:

$$\sum_{t=1}^{T_i} \delta_{i,t}^{\mathrm{cre}} = v(X_i) - v(B_i) - \delta_{i,0}^{\mathrm{cre}}. \tag{21}$$

This is the ordered analogue of efficiency for the fixed continuous attribution problem.

Prefix dummy If the encoded step block has zero marginal effect along the path, then it receives zero credit. Formally, if $\nabla_{x_{i,t}} v(\gamma_i(\alpha)) = 0$ for almost all $\alpha \in [0, 1]$, then $\delta_{i,t}^{\mathrm{cre}} = 0$. This statement does not require any special value of $v(B_i)$; the baseline value only sets the reference level in the completeness equation.

Linearity For two value functions v and w evaluated with the same baseline and path, let $h = av + cw$. Then $\nabla h = a\nabla v + c\nabla w$, and therefore

$$\delta_{i,t}^{\mathrm{cre}}(h) := a\delta_{i,t}^{\mathrm{cre}}(v) + c\delta_{i,t}^{\mathrm{cre}}(w). \tag{22}$$

Thus $\delta_{i,t}^{\mathrm{cre}}$ satisfies the continuous counterparts of efficiency, dummy, and linearity under temporal-causal constraints [33, 32]. □

B Additional Experiments

This appendix section collects supplementary material for the empirical part of the paper. We first compare baseline supervision regimes, define the evaluation metrics, then report the compute-cost comparison, additional analyses, and the full Best-of-N tables omitted from the main text.

B.1 Baseline Supervision Regimes

Table 7: Baseline supervision regimes. "Auto." marks automatically constructed process supervision; "No Ext." marks no external annotators, judges, or curated intermediate labels; "Marginal" marks whether step supervision accounts for marginal contribution.

Paradigm	Representative models	Auto.	No Ext.	Marg. contrib.
Human labels	Qwen2.5-Math-7B-PRM800K [45]; ReasonEval [39]	×	×	×
MC labels	Math-Shepherd [36]; RLHFlow [12]	✓	×	×
Outcome broadcast	OVM-Qwen3-4B [41]	✓	✓	×
First-error boundary	PQM-DeepSeek-zeta4-7B [19]	×	✓	×
Curated process	Skywork-PRM-7B [15]	-	-	×
Outcome-only credit	**SPRM-Qwen3-4B (ours)**	✓	✓	✓

The baselines in the main experiments differ not only in architecture and backbone but also in the source of their process supervision. Table 7 separates these differences along three axes that are central to our setting: whether the process signal is constructed automatically, whether it avoids external annotators, judges, or curated intermediate labels, and whether it explicitly assigns step credit by marginal contribution. Human-labeled and curated PRMs provide strong process signals but rely on external process supervision. Monte Carlo labeling and outcome broadcasting automate part of the labeling pipeline, but their targets do not isolate each step's marginal contribution to the final outcome. First-error boundary models avoid curated intermediate scores, yet they supervise boundary localization rather than a marginal credit allocation. SPRM is the only regime in this comparison that derives process labels from outcome-only trajectories while explicitly estimating marginal contribution.

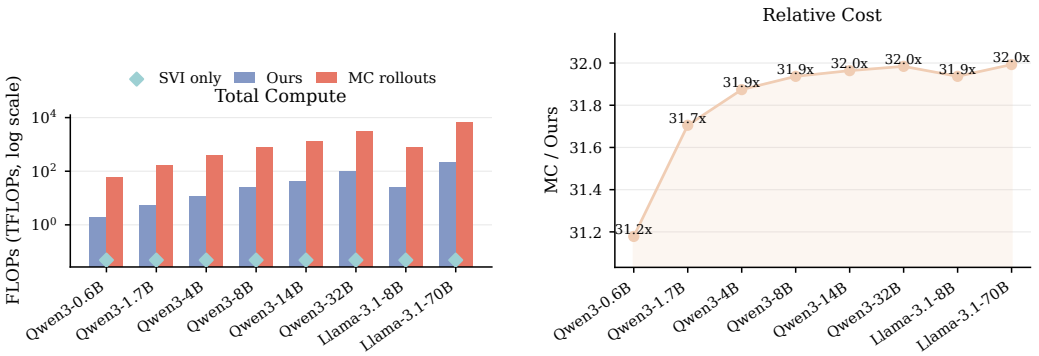
B.2 Evaluation Metrics

We summarize the evaluation metrics used for all benchmarks in the main text and appendix. ProcessBench [46] evaluates first-error localization. We report *Err*, *Cor*, and F1 to capture first-error localization, all-correct recognition, and their aggregate trade-off. PRMBench [31] evaluates fine-grained preference behavior through *Simplicity*, *Soundness*, and *Sensitivity*, which measure responses to redundant steps, factual or logical errors, and condition traps or multiple solution paths, respectively. For Best-of-N inference, we follow process-reward reranking [20] and report final-answer accuracy across problem difficulties and search ranges, averaged over 10 random seeds.

B.3 Computation Cost Comparison

We include an additional FLOPs comparison between SVI and MC-based step-credit estimation. The estimate uses the standard dense-LLM approximation $F_{\mathrm{LLM}} = 2PT$, where P is the active parameter count and T is the number of processed tokens. For SVI, the cost is one LLM trajectory pass plus BiLSTM SVI attribution; for MC, the cost is 32 repeated rollouts of the same prompt-and-solution length. We use 1024 prompt tokens, 512 solution tokens, 10 reasoning steps, final-layer step features of dimension 2560, a one-layer BiLSTM with hidden size 512, and 64 SVI integration steps. Figure 4 reports the resulting estimates. In the total-compute panel, the blue bars and red bars are complete method costs, while the cyan diamond is only the BiLSTM SVI overhead inside the blue bar.

As shown in Table 8, the BiLSTM SVI component is only 48.6 GFLOPs under this setting, so the SVI total is dominated by the single backbone trajectory pass. By contrast, MC scales linearly with the rollout budget, making it roughly $32\times$ more expensive across the tested backbones.



(a) Total FLOPs.

(b) Relative MC cost.

Figure 4: Estimated compute for SVI credit assignment versus MC rollout-based credit estimation. Blue and red bars show complete total FLOPs for ours and MC, respectively; the cyan diamond is the SVI-only overhead, not a separate complete method. MC uses 32 rollouts without shared prompt prefill.

Table 8: Representative FLOPs breakdown for Qwen3-4B.

Quantity	**FLOPs**	**Role in figure**
Ours total	12.3 TF	Blue bar
MC total	393 TF	Red bar
BiLSTM SVI overhead	48.6 GF	Cyan diamond
One BiLSTM forward	252 MF	Cost unit

B.4 Additional Analysis

B.4.1 Value Function Architecture Analysis

Figure 5 compares lightweight differentiable trajectory value function architectures by the downstream ProcessBench performance of the PRM trained from their induced step signals. For all non-BERT value functions, the input is the encoded sequence $X_i = [x_{i,0}, x_{i,1}, \ldots, x_{i,T_i}]$ used in Section 3.2, where $x_{i,0}$ is the *question_end* representation and $x_{i,t}$ is the representation of step $s_{i,t}$. For the BERT baseline, we instead aggregate SVI gradients over each step's token span at the embedding level. For the Transformer value function, we test max pooling, mean pooling, and attention-weighted fusion, and report the attention-weighted variant. For the MLP value function, we test max and mean pooling and report mean pooling.

The results suggest that compressed step-level representations are necessary: BERT obtains nearly zero F1, indicating that direct embedding-level attribution introduces mostly noisy and ineffective gradients. BiLSTM consistently outperforms LSTM across metrics, showing that bidirectional context helps model cross-step dependencies for final-reward prediction. MLP performs closest to BiLSTM, suggesting that feed-forward scoring preserves substantial information for gradient integration, while BiLSTM provides a better trade-off between information preservation and task-relevant sequence modeling.

B.4.2 Step Representation Analysis

In this section, we investigate three key questions regarding the step representation $x_{i,t} \in \mathbb{R}^d$: which single hidden layer provides the optimal representation, how many layers should be combined, and whether prefix differencing is beneficial. To determine which layers best capture step semantics, we conduct linear-probe experiments by training the trajectory value function on hidden states extracted from various layers. Furthermore, we train PRMs using varying numbers of hidden layers and evaluate the impact of differential versus raw inputs through qualitative case studies.

As illustrated in Figure 6(a) and (b), linear-probe experiments reveal that representations from deeper layers (those closer to the output) are significantly more effective for final-reward prediction by

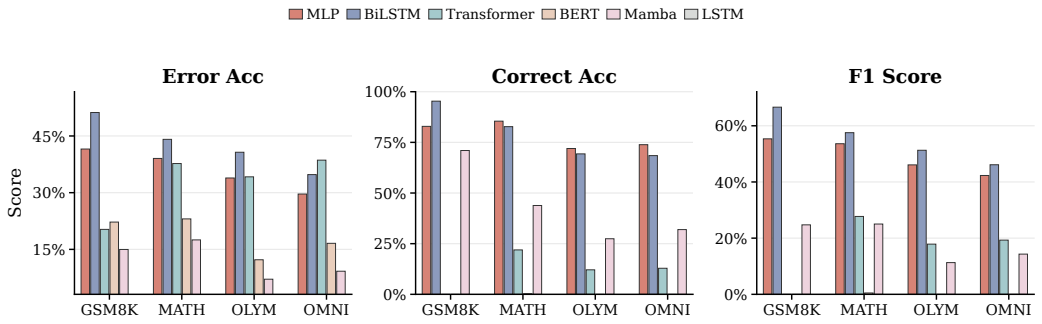
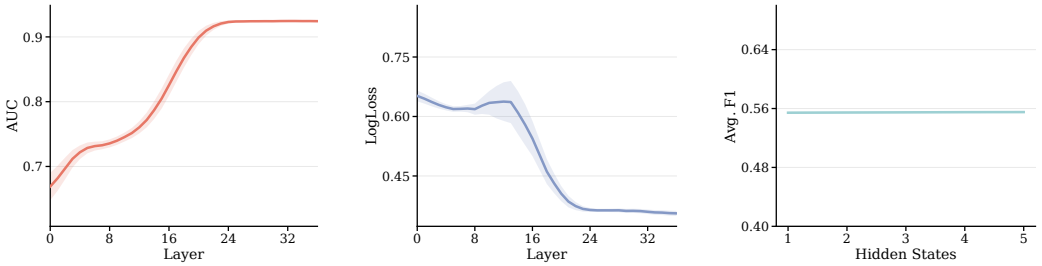


Figure 5: Trajectory value function architecture selection across all metrics. Different value function architectures are compared under a matched optimization setup.



(a) Layer-wise AUC. (b) Layer-wise log-loss. (c) F1 vs. hidden states.

Figure 6: Input feature selection: (a) AUC analysis, (b) log-loss analysis, and (c) F1 variation on ProcessBench.

the trajectory value function, with both AUC and log-loss converging to optimal levels in the final layers. Moreover, Figure 6(c) demonstrates that concatenating hidden states from multiple top layers yields no observable improvement in the downstream PRM F1 score. Consequently, we use the single final-layer hidden state for step representation, maximizing computational efficiency without compromising performance.

To address the differential-input question, we analyze a qualitative reasoning case where a critical mistake occurs at Step 1. Although subsequent steps maintain local computational consistency, they stem from a flawed premise and must ultimately be judged as incorrect. Our analysis reveals that raw hidden states better preserve global logical coherence by retaining full contextual history, whereas prefix differencing aggressively discards crucial cross-step dependencies.

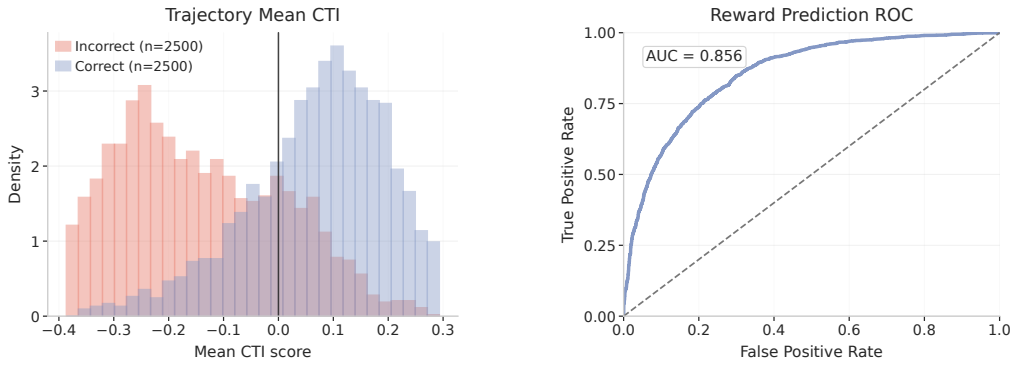
B.4.3 SVI Baseline Analysis

To choose the SVI baseline, we consider two options: (i) using a first-order forward-shifted sequence as the baseline, and (ii) using a single semantically empty vector b for all encoded positions, as in Section 3.2.

The motivation for option (i) is to capture incremental step contribution. In this case, for input $X_i = [x_{i,0}, \ldots, x_{i,T_i}]$, the baseline sequence is $B_i^{\text{shift}} = [b, x_{i,0}, \ldots, x_{i,T_i-1}]$, where b is the semantically empty vector used for the *question_end* position. Let $\delta_{i,t}^{\text{cre,shift}}$ denote the SVI credit computed with this shifted baseline. SVI completeness then gives:

$$\sum_{t=0}^{T_i} \delta_{i,t}^{\text{cre,shift}} = v(X_i) - v(B_i^{\text{shift}}). \tag{23}$$

For the trajectory value function, the objective is to predict final reward from step-level features, so $v(B_i^{\text{shift}})$ can be interpreted as the value prediction of a right-shifted trajectory. The resulting SVI



(a) Mean CTI score by reward group.

(b) ROC curve using mean CTI score.

Figure 7: CTI score analysis with two trajectory-level views.

signal therefore attributes steps relative to this shifted context, which is semantically misaligned with our intended notion of stepwise marginal contribution. For this reason, we do not adopt option (i).

For option (ii), summing per-step SVI signals gives:

$$\sum_{t=0}^{T_i} \delta_{i,t}^{\text{cre}} = v(X_i) - v(B_i), \tag{24}$$

where $B_i = [b, \ldots, b]$.

According to the SVI definition, each step attribution reflects its contribution relative to the baseline prediction. In [33], b is set to the all-zero vector for image attribution, with the goal of using a minimally informative state to avoid biased attributions. In process-reward modeling with binary outcome signals $(0/1)$, we define a minimally informative baseline as one producing near-neutral output, *i.e.*, $v(B_i) \approx 0.5$, so that attribution signs directly indicate whether a step is beneficial or harmful. Comparing an all-zero choice of b with random-noise choices, we find that the all-zero baseline yields outputs stably close to 0.5, whereas random baselines are less stable. We therefore adopt the all-zero vector for b, which preserves completeness and the semantics of feature-level SVI attribution, and best matches the Aumann-Shapley perspective.

B.4.4 CTI Score Analysis

To empirically validate the efficacy of the proposed CTI score $\delta_{i,t}^{\text{cti}}$, we analyze its trajectory-level distribution and predictive capacity. As defined in Section 3.3, $\delta_{i,t}^{\text{cti}}$ acts as a cross-trajectory inherent score, approximating the relative success tendency of specific semantic patterns via nearest-neighbor voting.

Figure 7(a) illustrates the distribution of the mean CTI score, aggregated over all steps within an individual trajectory, stratified by the final episodic return, with $n = 2500$ for each group. The empirical distributions exhibit a clear and substantial separation. Successful trajectories are predominantly characterized by a positive mean δ^{cti}, whereas unsuccessful trajectories consistently yield negative values. This observation aligns seamlessly with our theoretical premise: state representations associated with successful outcomes reliably retrieve high-reward neighbors from the cross-trajectory candidate pool Ω, confirming that $\delta^{\text{cti}} > 0$ effectively encodes a higher-than-average propensity for success.

To rigorously quantify the discriminative power of this score, Figure 7(b) presents the Receiver Operating Characteristic (ROC) curve for predicting the binary final reward $r_i \in \{0, 1\}$ solely using the trajectory-mean δ^{cti}. The estimator achieves an Area Under the Curve (AUC) of 0.856, demonstrating strong predictive performance significantly above random chance.

Collectively, these results substantiate the utility of our non-parametric voting estimator. By surfacing robust, cross-sample statistical constraints, the CTI score provides a highly discriminative empirical signal that complements the intra-trajectory step credits $\delta_{i,t}^{\text{cre}}$.

B.5 Full Best-of-N Results

This section reports the complete AceMath-RewardBench Best-of-N evaluation omitted from the main text for space. The full comparison includes dataset-wise PRM reranking results across all released reasoning-trajectory sets and candidate-pool sizes under the same sampling protocol. For SPRM, each entry reports the best accuracy over `soft`/`hard` score transformations and `min`/`mean` step aggregation. For PRM baselines, each entry reports the best accuracy over the same sorting variants when multiple variants are available. Pass@K and majority vote are reported as reference selection baselines above the PRM reranking models. We omit the `@1` column because all methods share the same greedy accuracy at N=1. In each remaining column, boldface marks the best score among reranking baselines and PRMs (*i.e.*, excluding the Pass@K upper bound).

Table 9: Complete Best-of-N accuracy on CollegeMath, Gaokao2023en, GSM8K, and MATH500.

Model	**@2**	**@4**	**@8**	**@16**	**@32**	**@64**
CollegeMath						
Pass@K (upper bound)	70.9	77.9	82.7	86.1	87.4	89.0
Majority vote	58.8	67.2	70.1	71.3	71.9	**72.0**
Llama3.1-PRM-D	59.6	60.0	60.1	59.4	57.9	55.9
Llama3.1-PRM-M	59.1	59.2	59.2	57.3	55.9	53.2
Math-Shepherd	60.5	60.8	60.8	60.2	60.6	57.0
Qwen2.5-Math-7B-PRM800K	59.1	59.2	59.2	59.2	59.2	59.2
Skywork-o1-PRM	61.0	63.1	63.9	64.6	63.1	61.0
ReasonEval-7B	58.5	58.6	58.6	58.6	56.9	54.5
PQM	59.2	59.6	59.8	59.6	58.9	57.8
OVM	58.6	58.8	58.6	58.2	57.1	56.0
SPRM	**66.1**	**69.4**	**70.5**	**71.5**	**72.0**	**72.0**
Gaokao2023en						
Pass@K (upper bound)	62.7	74.6	79.9	82.7	85.0	86.0
Majority vote	45.1	59.1	65.9	70.7	72.8	72.0
Llama3.1-PRM-D	45.0	44.8	44.9	44.9	44.9	44.9
Llama3.1-PRM-M	46.6	46.6	46.6	46.6	46.6	45.0
Math-Shepherd	48.9	49.6	49.6	49.6	49.6	49.6
Qwen2.5-Math-7B-PRM800K	46.3	46.5	46.5	46.5	46.5	46.5
Skywork-o1-PRM	52.9	57.8	59.8	61.6	63.0	64.1
ReasonEval-7B	45.9	46.0	46.0	46.0	46.0	46.0
PQM	47.8	49.5	50.3	50.6	49.8	49.0
OVM	47.0	47.8	48.2	47.6	47.1	46.8
SPRM	**59.1**	**68.4**	**72.1**	**73.0**	**73.2**	**73.0**
GSM8K						
Pass@K (upper bound)	94.7	98.1	99.2	100.0	100.0	100.0
Majority vote	87.1	94.2	96.2	96.9	**97.3**	**98.3**
Llama3.1-PRM-D	87.0	87.0	87.0	87.0	87.0	87.0
Llama3.1-PRM-M	87.9	88.1	88.1	88.1	88.1	88.1
Math-Shepherd	88.1	88.5	88.5	88.5	88.5	88.5
Qwen2.5-Math-7B-PRM800K	87.2	87.4	87.4	87.4	87.4	87.4
Skywork-o1-PRM	90.1	92.4	93.0	93.7	94.3	94.2
ReasonEval-7B	87.1	87.1	87.1	87.1	87.1	87.1
PQM	87.6	88.3	88.6	88.7	88.4	88.0
OVM	87.2	87.4	87.5	87.2	86.9	86.5
SPRM	**92.3**	**95.2**	**96.8**	**97.2**	97.2	97.0
MATH500						
Pass@K (upper bound)	81.5	91.1	93.9	95.6	96.4	97.0
Majority vote	65.8	79.4	86.2	88.5	**89.7**	**90.0**
Llama3.1-PRM-D	66.4	67.6	67.7	67.8	67.8	67.8
Llama3.1-PRM-M	66.4	66.8	66.8	66.8	66.8	66.8
Math-Shepherd	68.3	68.8	68.8	68.8	68.8	68.8
Qwen2.5-Math-7B-PRM800K	67.5	67.8	67.8	67.8	67.8	67.8
Skywork-o1-PRM	73.6	78.4	79.7	80.8	81.1	81.3
ReasonEval-7B	66.4	66.7	66.7	66.7	66.7	65.9
PQM	67.1	68.4	69.1	69.4	68.5	67.4
OVM	66.4	66.9	67.2	66.9	66.4	65.9
SPRM	**77.8**	**86.4**	**88.8**	**88.8**	89.0	**90.0**

Table 10: Complete Best-of-N accuracy on Minerva Math, MMLU-STEM, and OlympiadBench.

Model	@2	@4	@8	@16	@32	@64
Minerva Math						
Pass@K (upper bound)	41.4	53.8	62.8	68.9	72.1	75.0
Majority vote	28.6	38.2	44.5	**48.7**	**51.2**	**52.4**
Llama3.1-PRM-D	30.1	30.3	30.3	30.3	30.3	30.3
Llama3.1-PRM-M	29.3	29.5	29.5	29.5	29.5	29.5
Math-Shepherd	30.1	30.4	30.4	30.4	30.4	30.4
Qwen2.5-Math-7B-PRM800K	29.3	29.4	29.4	29.4	29.4	29.4
Skywork-o1-PRM	30.5	32.3	33.9	35.4	36.5	37.6
ReasonEval-7B	28.9	29.1	29.1	29.1	29.1	29.1
PQM	28.9	29.5	29.8	29.6	29.1	28.6
OVM	28.6	28.8	28.9	28.6	28.4	28.2
SPRM	**37.0**	**42.4**	**45.6**	45.3	46.2	46.0
MMLU-STEM						
Pass@K (upper bound)	76.4	90.9	97.3	99.4	99.8	100.0
Majority vote	53.8	69.4	76.2	81.1	83.9	85.8
Llama3.1-PRM-D	54.4	54.5	54.5	54.5	54.5	54.5
Llama3.1-PRM-M	54.4	54.6	54.6	54.6	54.6	54.6
Math-Shepherd	55.8	56.1	56.1	56.1	56.1	53.6
Qwen2.5-Math-7B-PRM800K	54.4	54.4	54.4	54.4	54.4	54.4
Skywork-o1-PRM	54.6	54.9	55.8	55.9	55.8	55.8
ReasonEval-7B	53.8	53.7	53.7	53.6	53.4	53.4
PQM	54.2	54.7	55.0	54.8	54.3	53.9
OVM	53.9	54.1	54.2	53.9	53.6	53.3
SPRM	**67.4**	**76.2**	**80.6**	**83.6**	**86.4**	**88.0**
OlympiadBench						
Pass@K (upper bound)	19.2	29.1	37.2	44.9	50.7	55.0
Majority vote	12.2	16.4	19.8	22.2	22.6	23.6
Llama3.1-PRM-D	11.1	11.4	11.0	9.6	8.7	7.6
Llama3.1-PRM-M	11.4	12.7	12.3	9.8	9.2	8.0
Math-Shepherd	13.8	14.6	14.6	14.6	14.6	14.6
Qwen2.5-Math-7B-PRM800K	12.2	12.2	12.2	12.2	12.2	12.2
Skywork-o1-PRM	12.4	12.7	13.2	14.3	15.0	17.3
ReasonEval-7B	12.0	12.3	11.8	12.1	11.8	9.1
PQM	12.4	12.8	13.0	12.8	12.4	11.8
OVM	12.2	12.4	12.5	12.3	12.2	11.7
SPRM	**16.1**	**19.6**	**20.5**	**23.1**	**24.4**	**26.0**

B.6 Same-Backbone Supervision Comparison

To isolate the effect of supervision, we retrain OVM, Monte Carlo, LLM-judge, and SPRM variants with the same Qwen3-4B-Instruct-2507 backbone on a shared, fixed subset of 20K Math-Shepherd trajectories. The remaining architecture, optimization, preprocessing, and ProcessBench evaluation settings are kept identical to the original implementation. The human-label reference uses the same backbone but a randomly sampled 20K subset of PRM800K, the available human process-labeled source, and therefore does not control for training data. Unlike the broader comparison in Table 2, which includes publicly available PRMs with different backbones and supervision regimes, this experiment controls the backbone and, except for the human reference, the training subset.

OVM broadcasts terminal outcome labels to all steps, whereas Monte Carlo uses rollout-estimated step labels. The LLM-judge baseline uses binary labels from DeepSeek-V3.2, which receives the problem, previous steps, and current step and answers only `Yes` or `No`. SPRM constructs marginal-contribution supervision from terminal outcome rewards without human step labels, external step judges, or rollout-derived step labels.

As shown in Table 11, SPRM achieves the highest F1 on all four ProcessBench splits, with an average F1 of 58.5 versus 57.0 for the strongest baseline, LLM-judge supervision. It exceeds outcome broadcasting and Monte Carlo supervision by 29.0 and 23.6 average-F1 points, respectively, using only 20K training trajectories. These results support the benefit of marginal-contribution supervision under a shared backbone and training subset.

Table 11: Same-backbone supervision comparison on ProcessBench (F1). All variants use Qwen3-4B-Instruct-2507 and 20K training trajectories. Math-Shepherd rows share the same subset; the human-label reference uses PRM800K instead. Boldface marks the best score in each column.

Supervision	Training source	GSM8K	MATH	OlympiadBench	OmniMath	Avg. F1
Outcome broadcast (OVM)	Math-Shepherd	8.6	46.5	32.9	30.1	29.5
Monte Carlo labels	Math-Shepherd	63.2	37.4	21.2	17.8	34.9
Human labels	PRM800K	66.6	55.2	38.4	31.4	47.9
LLM-judge labels	Math-Shepherd	68.2	58.5	51.0	50.4	57.0
SPRM (ours)	Math-Shepherd	**68.6**	**59.2**	**53.6**	**52.6**	**58.5**

B.7 Downstream Reinforcement Learning

We evaluate whether SPRM improves downstream policy optimization by training Qwen3-1.7B-Base on MATH and evaluating on MATH-500. We compare RLOO with terminal correctness rewards, RLOO + OVM with an outcome-broadcast process reward model, and RLOO + SPRM with our marginal-contribution process reward model. All three settings share the initial policy and training setup. OVM and SPRM use the same reward-model backbone and the same min-form reward integration with RLOO, allowing a controlled comparison of their process supervision signals. We use the default min-form hyperparameters for both process-reward variants.

We optimize the policy with AdamW using a constant learning rate of 5×10^{-6}, weight decay 0.01, and LoRA rank 64 with alpha 128. Each training batch contains 32 prompts with eight rollouts per prompt. We use a policy-update mini-batch size of 32, one policy-optimization epoch per rollout batch, and a PPO clipping ratio of 0.2. Training rollouts use temperature 1.2 and top-p 1.0, with prompt and response lengths each capped at 1,024 tokens.

We repeat RL training with three independent random seeds for each method. For each run, we evaluate mean@8 accuracy, averaging correctness over eight sampled responses per problem. Table 12 reports the mean and standard deviation across independent RL training seeds at each reported training step. The shared initial policy achieves 55.0% mean@8 accuracy before RL training.

Table 12: Downstream RL results on MATH-500 with Qwen3-1.7B-Base trained on MATH. Entries report mean@8 accuracy (%), as mean $\pm$ standard deviation across three independent RL training seeds at each training step. OVM and SPRM use the same min-form reward integration. Boldface marks the highest mean in each row.

Training step	RLOO	RLOO + OVM	RLOO + SPRM
25	59.4 ± 1.7	54.2 ± 1.1	**60.0** ± 1.5
50	63.0 ± 0.5	57.2 ± 0.8	**63.4** ± 1.1
75	63.6 ± 0.6	57.8 ± 1.2	**66.2** ± 0.7
100	64.4 ± 0.8	57.4 ± 0.8	**68.0** ± 0.9
125	65.6 ± 0.7	56.4 ± 1.9	**69.2** ± 0.7
150	65.2 ± 1.2	58.2 ± 1.3	**70.0** ± 0.8

At training step 150, RLOO + SPRM reaches 70.0% mean@8 accuracy, exceeding RLOO by 4.8 percentage points and RLOO + OVM by 11.8 points. RLOO + SPRM has the highest mean accuracy at every reported training step. Its advantage over OVM under the same reward integration supports the usefulness of marginal-contribution supervision for downstream policy optimization in this setting.

C Implementation and Reproducibility Details

This appendix section collects implementation and runtime details for reproducing the reported runs. The empirical analyses and additional result tables are kept separately in Appendix B.

C.1 Component Implementation

Our implementation consists of three components: trajectory value function training, step-level attribution construction, and final PRM training. Unless otherwise stated, all components use Qwen3-4B-Instruct-2507 as the backbone.

Trajectory value function training We randomly sample 220K trajectories from Math-Shepherd and extract final-layer hidden states for the question boundary and reasoning steps as encoded positions. The question-boundary representation is kept as the zeroth encoded position, so each trajectory is represented as $[x_{i,0}, x_{i,1}, \ldots, x_{i,T_i}]$, where $x_{i,0}$ is *question_end*. Each representation has dimension $d = 2560$, matching the final hidden size of Qwen3-4B-Instruct-2507. The trajectory value function is a single-layer bidirectional LSTM with hidden size 512 in each direction, followed by a linear prediction head that outputs one trajectory logit. It is trained with $\mathrm{BCEWithLogitsLoss}$ on the binary final reward, using post-padding and packed sequences so that padded positions are ignored. We train with batch size 128 and learning rate 1×10^{-4}, with early stopping using patience 5, and split the data into train/validation/test sets with ratio $0.8/0.1/0.1$.

Step-level attribution construction We compute SVI step attribution with an all-zero baseline and 64 integration steps, using the trajectory value function probability output as the attribution target. For each trajectory, the encoded sequence is treated as a single differentiable input, and each step credit is obtained by summing the SVI attribution over the hidden dimensions of the corresponding step representation. To construct the step-credit supervision mask, we compare the sign of this SVI credit with the sign of the model-counterfactual consistency score $R_{i,t}^{\mathrm{count}}$. For the local interaction window in Equation (6), we use $w = 3$ and $\beta = 0.5$. The label for the step-credit head is used only when the two signs agree; the gate threshold is 0, so no additional magnitude filtering is applied. The cross-trajectory inherent score is built through semantic nearest-neighbor retrieval across trajectories. We L2-normalize all step representations and retrieve neighbors by cosine similarity, excluding steps from the same trajectory to avoid outcome leakage. For each step, we search over 400 candidate neighbors, keep the top 50 valid neighbors, and compute similarity-weighted reward statistics using ReLU-based weights. The CTI score is the resulting neighborhood success rate minus the dataset base success rate.

PRM training The constructed step-credit and CTI signals are then used to supervise the final PRM. We insert a special `<ST_END>` token after the question boundary and after every reasoning step, and train both value heads only at these boundary positions. The first head predicts the binarized SVI step-credit label under the R^{count}-consistency mask, while the second head predicts the binarized CTI label whenever the CTI score is finite. Both labels use threshold 0, assigning positive scores to label 1 and nonpositive scores to label 0. The total objective is the sum of the two masked BCE losses with equal weights. We perform LoRA fine-tuning on the backbone with target modules `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. Training uses maximum context length 4096, runs for 1 epoch with learning rate 5×10^{-5}, and uses global batch size 32.

C.2 Run Details and Environment

This subsection records the runtime environment for the reported reproducible runs. Hyperparameters and component-specific training choices are reported above.

Execution environment The reported runs use Linux `5.15.0-164-generic`, Python `3.11.14`, CUDA `12.8` as used by PyTorch, PyTorch `2.9.1`, Transformers `4.56.1`, Accelerate `1.6.0`, DeepSpeed `0.18.3`, PEFT `0.18.0`, and TRL `0.11.4`. The dependency constraints require Python $\geq 3.11, < 3.12$.

Hardware and launch Experiments are run on two NVIDIA L20 GPUs with 45 GB memory each. Trajectory value function training, step-level attribution construction, and final PRM training use `torchrun -nproc_per_node 2 -master_port $MAIN_PORT`, where `MAIN_PORT` is selected automatically from 29500–29649 to avoid port conflicts.

Logging and deterministic settings Weights & Biases logging is enabled by default unless `WANDB_DISABLED` is set. Trajectory value function training and final PRM training use random seed 42; step-level attribution construction uses `STAGE2_DG_SAMPLE_SEED` $= 1234$ for CTI trajectory sampling. We fix and report these seeds, but do not claim bitwise deterministic reproduction because multi-GPU execution, cuDNN, DeepSpeed, and fused AdamW may still introduce small non-determinism.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the core claim of deriving step-level supervision from outcome-only trajectories, and the conclusion keeps the same scope in light of the reported ProcessBench, PRMBench, and Best-of-N results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The conclusion explicitly states that the current method assumes verifiable outcome rewards and identifies reward modeling for tasks whose answers cannot be reliably verified as an important direction for future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The main text states the two lemmas and cross-references the appendix, which defines the notation, lists assumptions A1–A4, and provides the corresponding proofs and axiomatic comparison.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The experiments section describes datasets, baselines, metrics, and evaluation protocol, while Appendix C provides implementation details for value-function training, attribution construction, retrieval, PRM training, runtime environment, distributed launch, hardware, logging, and seeds.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We provide a public code repository at `https://github.com/by0214/sprm-open`, while the paper and appendix describe the datasets, baselines, metrics, evaluation protocol, implementation details, runtime environment, hardware, logging, and seeds needed to reproduce the main experimental results.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The experiments section and Appendix C specify the benchmarks, training data, backbone, evaluation protocol, data split, SVI settings, retrieval configuration, PRM training hyperparameters, runtime environment, and distributed launch details.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: The main benchmark tables do not report error bars or formal significance tests, although Best-of-N results are averaged over fixed random-seed reranking runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The appendix reports the two-GPU L20 setup, distributed launch details, runtime environment, and FLOPs comparison between SVI and MC rollout-based credit estimation.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work is a methodological study on process reward modeling using existing benchmark trajectories and does not involve deception, human-subject experimentation, or release practices that intentionally conflict with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The conclusion discusses both positive impacts, such as reducing dependence on costly external judges, and negative impacts, such as strengthening automated reasoning systems in settings where persuasive errors can be harmful.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a new high-risk generative model or scraped dataset as part of the current submission.

    Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper credits the benchmark datasets, models, and prior systems used in the work through the corresponding citations.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [N/A]

Justification: The current submission does not release a new dataset, code package, or model artifact as a documented public asset.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve human-subject research requiring IRB review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are used only as existing experimental backbones or representation sources and are not an important, original, or non-standard component of the proposed core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.