RAWR: REWARD ASSIGNMENT WITHOUT ROLLOUTS IN VERIFIABLE DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding and evaluating multi-step reasoning in Large Language Models (LLMs) at the level of individual steps remains a key challenge. Process Reward Models (PRMs) provide a solution by scoring each step, enabling fine-grained supervision and improved reliability. However, training them requires costly human annotation or computationally intensive rollout-based labeling. To solve this, we introduce Reward Assignment Without Rollouts (**RAWR**), a scalable method for automatically labeling the quality of individual reasoning steps in any verifiable domain. Its step score, net information gain (NetIG), improves upon single-reference Information Gain (IG) by comparing the most-supported correct answer against the most-supported incorrect one, yielding a robust signal even for long and structured outputs like code and SQL, where IG fails. We show that the signal produced by **RAWR** correlates with human judgments of step quality, and we apply **RAWR** labels to train PRMs that achieve the best average best-of-K accuracy across eight benchmarks spanning mathematics, code generation, text-to-SQL, and scientific QA. Crucially, **RAWR** generates no rollouts, cutting labeling complexity to $\mathcal{O}(N)$ and making it up to $77\times$ cheaper than rollout-based methods at comparable label quality, which makes large-scale process supervision practical.

1 INTRODUCTION

Reasoning models such as OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) solve hard multi-step problems by generating long Chain-of-Thought (CoT) traces and scaling test-time compute (Snell et al., 2024). Yet this reasoning remains unreliable at the step level: in mathematics (Hendrycks et al., 2021b), code generation (Zhuo et al., 2024), text-to-SQL (Li et al., 2024), and scientific or medical QA (Xu et al., 2025; Jin et al., 2019; Royer et al., 2024), a single flawed step can derail the solution, and even correct final answers often rest on erroneous intermediate steps (Zheng et al., 2025).

Reward models (RMs) have been central to the development of conversational Large Language Models (LLMs) (Ouyang et al., 2022), enabling alignment with human preferences and more reliable generation. They can be broadly categorized as outcome reward models (ORMs) (Cobbe et al., 2021; Shi et al., 2022) and Process Reward Models (PRMs) (Lightman et al., 2023; Wang et al., 2024a;b; Zhang et al., 2025). ORMs assign a single score to the final output, providing coarse-grained evaluation that suffices when only correctness matters. In contrast, a PRM takes a step-by-step answer to a question and grades each individual step. This fine-grained evaluation allows for more nuanced feedback and facilitates deeper inspection of where and how reasoning may go astray. By capturing the quality of individual steps, we improve the ability of the reward model to identify reasoning errors. This also enhances interpretability by identifying which steps contribute positively or negatively to the final answer. PRMs are particularly useful in tasks requiring transparency and multi-step reasoning, such as programming and scientific QA.

Recent advances in test-time scaling and reinforcement learning have made research on PRMs more relevant than ever. A wide range of methods, from simple re-ranking and answer selection (Wei et al., 2022; Stiennon et al., 2020) to more sophisticated approaches like tree-based search (Yao et al., 2023), rely on accurate estimations of the value of individual reasoning steps, making them well-suited to benefit from the fine-grained supervision offered by PRMs, and when training using reinforcement learning, guiding the model with a PRM rather than an ORM leads to higher performance (Wang

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

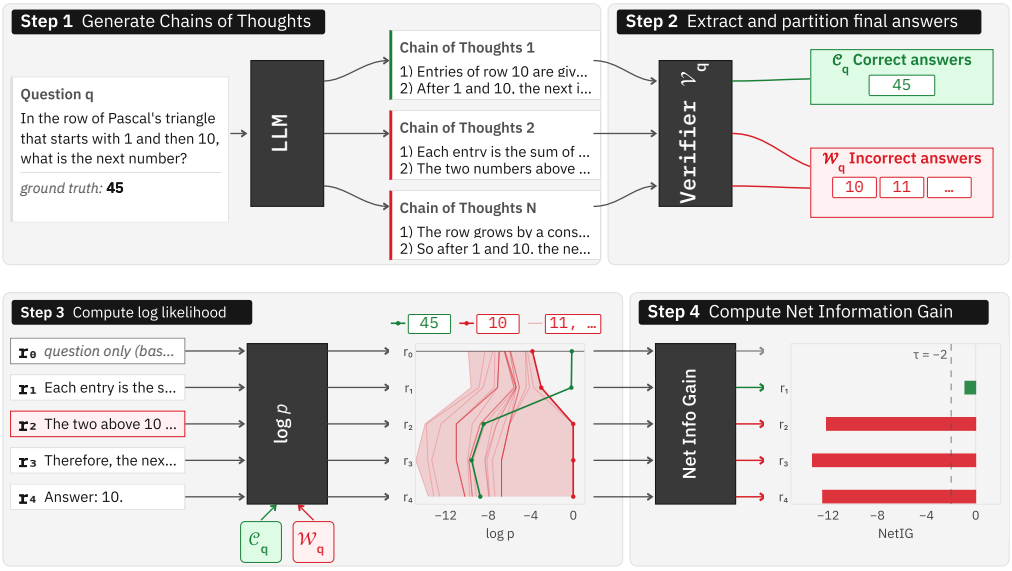

Figure 1: Overview of **RAWR**. **(1)** An LLM samples K CoTs for question q. **(2)** A verifier $\mathcal{V}_q$ partitions them into correct ($\mathcal{C}_q$) and incorrect ($\mathcal{W}_q$) sets. **(3)** At every prefix $r_0, \ldots, r_n$, we score each candidate answer and keep the max over $\mathcal{C}_q$ and $\mathcal{W}_q$. **(4)** The step score, NetIG, is the change in the correct-incorrect gap relative to r_0, thresholded at τ to produce binary step labels.

et al., 2024a). Prior work has largely focused on mathematical reasoning tasks (Lightman et al., 2023; Wang et al., 2024a), leaving domains such as code generation mostly unexplored. Expanding the range of tasks where a PRM can be effectively applied is crucial; unfortunately, gathering a large enough dataset to train such a model is a difficult and expensive endeavor, often requiring substantial human labeling effort. This hinders the scalability and adoption of PRMs, motivating the need for more efficient data labeling methods.

In this work, we introduce Reward Assignment Without Rollouts (**RAWR**), a method for automatically measuring the quality of individual reasoning steps in CoT traces. It requires only an LLM and a binary validator, so it applies to any verifiable domain: the LLM samples a set of solutions (Figure 1, step 1), and the validator partitions them into correct and incorrect answers (step 2), yielding a Monte Carlo approximation of the correct (*resp.* incorrect) answer space. At every reasoning prefix, we compute the log-likelihood of each candidate answer (step 3) and measure how the step shifts the model toward the correct answer space relative to the incorrect one (step 4). We call this signal the net information gain (NetIG) of the step, and threshold it into a binary reward. Unlike rollout-based methods, **RAWR** never samples continuations from intermediate prefixes: the K solutions it draws are the traces being labeled, and the rest is a forward pass over them.

The benefits of this contrastive formulation are many. First, it makes the labeling signal robust to tasks with long or compositional outputs, where simpler Information Gain (IG) measures degrade. Second, its technical requirements are minimal: **RAWR** is applicable to any autoregressive language model and any task equipped with a binary validator, it requires no human annotation, no generation beyond the sampled traces themselves, and no task-specific design choices beyond the validator itself. Another benefit of **RAWR** is its complexity: $\mathcal{O}(N)$ in the number of reasoning steps through KV-cache reuse, making it 23× cheaper to run than OMEGAPRM (Luo et al., 2024), the cheapest rollout-based method, once output tokens are priced at the commercial rate (Section A.1), making it efficient and scalable. In this paper, we apply **RAWR** labels to train PRMs, but the step-quality scores are general-purpose to any application requiring fine-grained reasoning evaluation. We summarize our contributions as follows:

1. **Rollout-free reward assignment for reasoning steps.** We introduce **RAWR**, a method for scoring individual CoT steps that achieves $\mathcal{O}(N)$ complexity.

2. **Strong PRMs across diverse domains.** We apply **RAWR** to train PRMs that achieve the highest average best-of-K accuracy across eight benchmarks spanning mathematics, code generation, text-to-SQL, and scientific QA.
3. **Step labels that track human judgments.** **RAWR** scores increase monotonically with expert step labels on PRM800K and localize the first erroneous step in ProcessBench traces.

More broadly, our work demonstrates that step-level supervision does not require rollouts: a single scoring pass over sampled reasoning traces is enough to train competitive PRMs, reducing the barrier from months of expert annotation or quadratic rollout budgets to a forward pass. By extending the scope of process supervision to code and SQL domains, we show that fine-grained step-level feedback benefits any task requiring multi-step reasoning, a setting common to most real-world applications of LLMs. Beyond PRM training, **RAWR** labels open avenues for reward shaping in reinforcement learning, and interpretability of multi-step reasoning.

2 Related work

Verifier-guided reasoning. Sampling multiple reasoning traces and selecting the best with a verifier improves LLM accuracy when single-shot accuracy is low (Cobbe et al., 2021). Lightman et al. (2023) scaled this to step-level verification by creating PRM800K, 800K expert-labeled reasoning steps for MATH (Hendrycks et al., 2021b); training a step classifier on it substantially improved best-of-K selection and established PRMs as a tool for reasoning verification. The reliance on expert annotators, however, makes this expensive and hard to extend to new domains.

Reducing the cost of step-level labels. To avoid human annotators, MathShepherd (Wang et al., 2024a) generates rollout completions from each reasoning prefix and measures the fraction reaching the correct answer, but requires $\mathcal{O}(N^2)$ token processing. OmegaPRM (Luo et al., 2024) reduces this to $\mathcal{O}(N \log N)$ with binary search over the first incorrect step, but still requires substantial computation. Our method achieves $\mathcal{O}(N)$ complexity (Appendix A) with no additional generations, only with a single scoring pass over existing reasoning traces. Table 1 summarizes these methods.

Table 1: Comparison of step-level supervision methods. **RAWR** requires no rollouts and achieves $\mathcal{O}(N)$ complexity. Relevant prior methods have been validated on mathematics only.

Method	Label type	Complexity	Rollout-free	Cross-domain
PRM800K (Lightman et al., 2023)	Human expert	—	✓	✗
MathShepherd (Wang et al., 2024a)	Rollout	$\mathcal{O}(N^2)$	✗	✗
OmegaPRM (Luo et al., 2024)	Rollout + binary search	$\mathcal{O}(N \log N)$	✗	✗
QwenPRM (Zhang et al., 2025)	Rollout + consensus filtering	$\mathcal{O}(N^2)$	✗	✗
RAWR (ours)	Information Theoretic	$\mathcal{O}(N)$	✓	✓

Deriving process signals from outcome supervision. An alternative approach avoids step labels entirely by repurposing outcome-level supervision. Yu et al. (2024) introduce outcome-supervised value models that assess the potential of incomplete reasoning paths, reframing guided decoding as value estimation. Yuan et al. (2024) derive implicit process rewards from the log-likelihood ratio between a policy and a reference model, enabling PRM-like behavior from outcome-only training, and PRIME (Cui et al., 2025) uses this implicit PRM to provide dense rewards during reinforcement learning. While these methods eliminate step-level annotation, their step scores are implicit proxies learned from outcome labels rather than direct measures of step quality; in our experiments, OVM and ImplicitPRM trail **RAWR** by 5.7 and 11.5pp in average best-of-K accuracy (Table 2).

Information-theoretic analysis of reasoning. Recent work has applied information-theoretic tools to understand and evaluate CoT reasoning. Chen et al. (2023) evaluate free-text rationales using conditional V-information, measuring the label-relevant content a rationale adds beyond the input. Ton et al. (2024) analyze information flow through reasoning traces, quantifying how intermediate steps influence the likelihood of correct answers. Li et al. (2026) introduce step entropy as a measure of redundancy in reasoning, showing that not all steps contribute equally to predictive power. These approaches use information measures primarily for *analysis*, scoring existing rationales or diagnosing failures. **RAWR** instead operationalizes a contrastive IG signal as a *labeling* mechanism, directly producing training data for competitive PRMs.

3 PRELIMINARIES

We formalize the objects **RAWR** operates on (steps 1-2 of Figure 1): CoT traces sampled for a question q and their partition into correct and incorrect answer sets $\mathcal{C}_q, \mathcal{W}_q$. We then review IG, the single-reference predecessor of our signal, and identify its failure mode on long or structured answers, which **RAWR** is designed to fix.

3.1 PROBLEM SETUP AND NOTATION

Let p_θ denote an autoregressive language model that defines a distribution over token sequences via $p_\theta(x_t \mid x_{<t})$. A CoT is a sequence of intermediate steps $R = (r_1, \ldots, r_N)$, where each step r_i is a subsequence of tokens, and we write $R_{1:i} = (r_1, \ldots, r_i)$ for the prefix up to step i. The collection of traces for question q is $\mathcal{R}_q = \{R_1, \ldots, R_K\}$.

From each CoT we extract a final answer $y = (y_1, \ldots, y_{|y|})$, which is a subsequence of the tokens of the last step, i.e. $\{y_1, \ldots, y_{|y|}\} \subseteq \{r_{N,1}, \ldots, r_{N,|r_N|}\}$, yielding a set of candidate answers $\mathcal{Y}_q = \{y_1, \ldots, y_K\}$. We write $y_{<t}$ for the first $t-1$ tokens of y. A domain-specific validator $V_q : \mathcal{Y} \to \{0, 1\}$ determines whether a candidate answer is correct, and we partition:

$$\mathcal{C}_q = \{\, y \in \mathcal{Y}_q \mid V_q(y) = 1 \,\}, \qquad \mathcal{W}_q = \{\, y \in \mathcal{Y}_q \mid V_q(y) = 0 \,\}. \tag{1}$$

3.2 INFORMATION GAIN

A natural approach to evaluating the quality of a reasoning step is to measure how much it increases the model's confidence in the correct answer (Ton et al., 2024). Given a partial reasoning prefix $R_{1:i}$, we define the *information* that p_θ assigns to an answer y at step i as the total log-likelihood:

$$\mathcal{I}_i(y) = \sum_{t=1}^{|y|} \log p_\theta(y_t \mid q,\, R_{1:i},\, y_{<t})\,. \tag{2}$$

When no reasoning is provided (i=0), this reduces to $\mathcal{I}_0(y) = \sum_{t=1}^{|y|} \log p_\theta(y_t \mid q, y_{<t})$. The ***IG*** at step i for a reference answer $y^\star$ is then:

$$\mathrm{IG}_i = \mathcal{I}_i(y^\star) - \mathcal{I}_0(y^\star), \tag{3}$$

measuring how much the partial reasoning $R_{1:i}$ improves the model's support for $y^\star$ relative to the no-reasoning baseline. We subtract the no-reasoning baseline $\mathcal{I}_0$ rather than the preceding step $\mathcal{I}_{i-1}$ because the cumulative reference is invariant to how the model partitions reasoning into steps: finer step granularity reduces per-step likelihood changes, which would make threshold-based labeling sensitive to formatting choices rather than reasoning quality.

Limitations of single-reference IG A critical limitation of IG is its dependence on a *single* reference formulation $y^\star$. For mathematical tasks, this is benign: a numeric answer like "42" admits essentially one formulation. For structured outputs such as code or SQL, however, the space of semantically equivalent correct answers can grow rapidly with answer length. Variable naming choices (e.g., `result` vs. `res` vs. `ans`), equivalent expressions (e.g., `x != None` vs. `x is not None`), formatting conventions, and library alternatives each multiply the number of valid formulations. The number of valid formulations can thus grow combinatorially with answer length L, though correlations between choices reduce it in practice; IG degrades as it grows (Figure 2, right).

4 METHOD

RAWR replaces IG's single reference answer with a contrastive signal between the correct and incorrect answer sets (steps 3-4 of Figure 1). We motivate its two design choices (aggregation over formulations and correct-incorrect contrast) and give its full definition.

4.1 AGGREGATING OVER ANSWER FORMULATIONS

The limitation identified in Section 3.2 points to a natural target quantity: rather than measuring the model's support for a single reference answer, we want to estimate $\log p_\theta(\text{correct} \mid q, R_{1:i})$, the

model's total log-probability of producing *any* correct answer. Since the model assigns probability to individual token sequences rather than to equivalence classes, this quantity decomposes as:

$$\log p_\theta(\text{correct} \mid q, R_{1:i}) = \log \sum_{y \in \mathcal{C}_q} p_\theta(y \mid q, R_{1:i}). \tag{4}$$

Evaluating Eq. (4) exactly is intractable since $\mathcal{C}_q$ may be large, so we must estimate it from the finite set of K sampled answers.

Note that we have access to individual log-likelihoods $\mathcal{I}_i(y) = \log p_\theta(y \mid q, R_{1:i})$ for each sampled answer y. We can use these sampled log-likelihoods to estimate the target quantity. However, this estimator accounts only for formulations that happen to appear among the K samples, missing the tail of the equivalence class. For structured outputs such as code or SQL, where the number of valid formulations can be large, K samples provide incomplete coverage and the estimator remains unstable as each newly discovered formulation increases the sum.

We instead propose to use the $\max$ operator: $\max_{y \in \hat{\mathcal{C}}_q} \mathcal{I}_i(y)$. This formulation has low sample complexity: it requires only that the sample contains *one* instance of a high-probability formulation to stabilize, rather than covering the full support of the equivalence class. Formally, the $\max$ provides a lower bound on the target quantity:

$$\max_{y \in \hat{\mathcal{C}}_q} \mathcal{I}_i(y) \;\le\; \log \sum_{y \in \mathcal{C}_q} p_\theta(y \mid q, R_{1:i}), \tag{5}$$

and the gap depends only on the concentration of p_θ around its preferred formulation, not on the size of the equivalence class. When the model concentrates most of its correct-answer mass on a small number of preferred phrasings, the $\max$ closely tracks the true class probability regardless of how many low-probability alternatives exist. We validate this choice empirically in Appendix D: $\max$ outperforms summing over alternative answers (82.0% vs. 81.6% balanced accuracy).

4.2 Contrastive step scoring

Building on the $\max$ estimator, we define the *Net Information* at step i as the gap between the best-supported correct and incorrect answers:

$$\text{NetInfo}_i = \underbrace{\max_{y \in \mathcal{C}_q} \mathcal{I}_i(y)}_{\substack{\text{most-supported} \\ \text{correct answer}}} - \underbrace{\max_{y \in \mathcal{W}_q} \mathcal{I}_i(y)}_{\substack{\text{most-supported} \\ \text{incorrect answer}}}. \tag{6}$$

The **net information gain** (NetIG) at step i, the score **RAWR** assigns to each step, is then the net information relative to baseline:

$$\text{NetIG}_i = \text{NetInfo}_i - \text{NetInfo}_0. \tag{7}$$

By subtracting the pre-reasoning baseline, NetIG_i measures the contribution of step r_i: positive values indicate the prefix shifts the model toward a correct outcome, while negative values indicate it introduces or reinforces errors. Anchoring at r_0 rather than r_{i-1} captures reasoning that drifts into error gradually over several steps rather than at a single step (see Appendix D for the ablation). This matches how step-level supervision is usually defined: rollout methods condition on the prefix $R_{1:i}$ and human `PRM800K` annotations judge each step in the context of its predecessors. Figure 1 illustrates the full pipeline on a concrete example.

4.3 Label assignment

To train PRMs, we require binary labels. We thus project the continuous NetIG signal by thresholding:

$$\ell_i = \begin{cases} 1 & \text{if } \text{NetIG}_i > \tau_q, \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

where τ_q is selected by optimizing balanced accuracy of trace-level labels (the product of step labels for each trace), ensuring the resulting binary labels are not biased toward positive or negative steps. Because the final answer y is a subsequence of the last step r_N, we restrict the prefix sweep to $i < N$ to select the threshold, so the signal cannot degenerate by relying only on the last step.

5 EXPERIMENTS

We evaluate **RAWR** along two complementary axes. First, we assess label quality directly by measuring how well **RAWR** scores separate correct from incorrect reasoning steps across domains with varying answer lengths (Section 5.2). Second, we evaluate the labels indirectly through downstream PRM performance on best-of-K selection across diverse benchmarks, including out-of-distribution settings (Section 5.3-Section 5.4).

5.1 EXPERIMENTAL SETUP

Chain-of-thought generation. We prompt **Ministral-8B-Instruct-2410** with nucleus sampling (T=1.0, top-p=0.95) to generate K=32 completions per problem, formatted with `[STEP]` delimiters for consistent parsing. The mid-sized generator ensures a mix of correct and incorrect completions for contrastive scoring, and K=32 provides diverse formulations for the max operator (label accuracy plateaus by $K \approx$ 20–30; Appendix D). All 32 answers are retained for **RAWR** scoring. For PRM training, we remove unparseable answers, drop problems solved by all candidates, and subsample 8 traces per problem to control the correct/incorrect ratio, keeping a correct trace when available. Answers are validated with domain-specific tools: `math-verify` (Hugging Face, 2024) for mathematics, unit-test execution for Python, and execution accuracy for SQL. The full scoring prompt template is given in Appendix F. Full details are in Appendix B.

Datasets. Our training set spans four domains (189K mathematics, 36K text-to-SQL, 5704 Python, and 1000 medical QA problems) in total, we have 232K samples and 1.49M candidate answers. We evaluate on eight benchmarks covering the same domains plus `AIME 23-25` and the out-of-distribution `UGPhysics` (Xu et al., 2025). For the coding benchmarks we hold out a training subset (64 from `HumanEval`, 640 from `BigCodeBench`) and evaluate on the remainder. All results use best-of-K selection (K=32), where PRM scores are the product of step-level predictions and ORM scores the final classification probability. Full dataset details are in Appendix B.

PRM training. Given step-level labels, we train PRMs on the Ministral 3 8B and 14B backbones to predict correctness at each reasoning step. The input interleaves the question and steps with a delimiter token $\langle s_{\text{req}} \rangle$:

$$x = \big(q,\ r_1,\ \langle s_{\text{req}} \rangle,\ r_2,\ \langle s_{\text{req}} \rangle,\ \ldots,\ r_N,\ \langle s_{\text{req}} \rangle\big),$$

and at each delimiter the model classifies the step via two reserved tokens $\langle s_{\text{POS}} \rangle, \langle s_{\text{NEG}} \rangle$, trained with binary cross-entropy averaged over steps (batch size 128, AdamW, learning rate $1\text{e}{-6}$, max length 8,192, two epochs). ORM baselines are identical but classify only after the full trace.

5.2 LABELING QUALITY AND EFFICIENCY

A central prediction of the analysis in Section 3.2 and Section 4.1 is that single-reference IG should degrade on tasks with long, structured answers, while **RAWR** should remain robust. We now validate this prediction empirically.

RAWR provides more reliable labels than IG. We evaluate labeling quality by measuring the balanced accuracy of CoT-level predictions. Across all datasets, **RAWR** yields better accuracy than IG (Figure 2, left). The improvement is especially pronounced on code generation (Python and SQL) tasks, precisely the domains where the theoretical analysis predicts IG degradation due to long, compositional outputs.

RAWR scores correlate with human judgments. To validate that **RAWR** captures genuine step quality rather than superficial statistical artifacts, we compare its scores against human annotations from `PRM800K` (Lightman et al., 2023), which provides expert step-level labels (positive, neutral, negative) for mathematical reasoning traces. Mean NetIG scores increase monotonically across human label categories: -0.38, $+0.52$, and $+1.70$, confirming that the information-theoretic signal aligns with human judgments of reasoning quality. This is in line with previous findings that information gain correlates with human step-level evaluations (Ton et al., 2024), and extends this validation to the NetIG estimator.

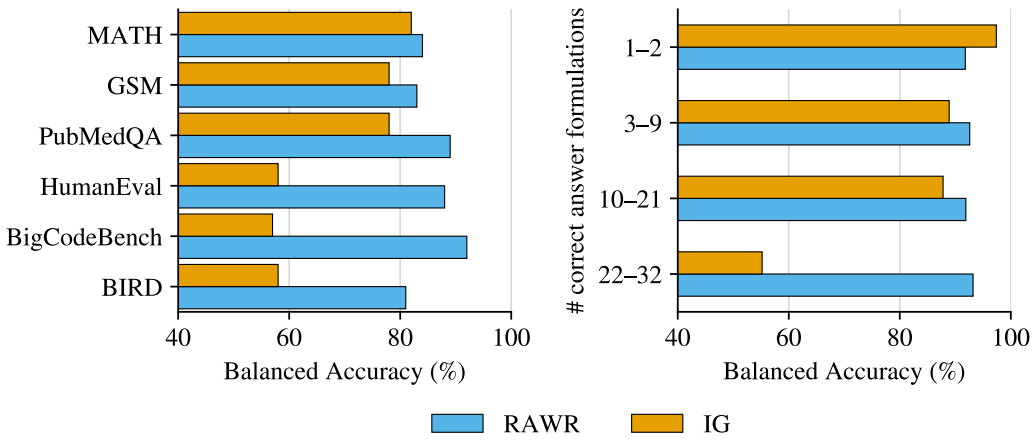

Figure 2: **Left:** Balanced accuracy of CoT-level labels for IG vs. **RAWR** across domains. **Right:** Accuracy stratified by the number of distinct correct answer formulations. IG degrades as formulation diversity increases, while **RAWR** remains stable.

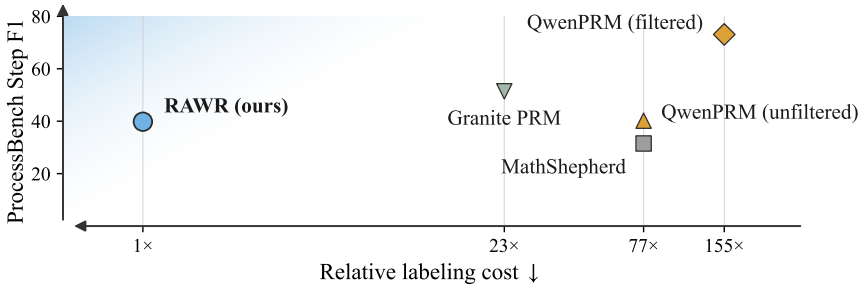

Figure 3: `ProcessBench` Step F1 against labeling cost relative to **RAWR** (Section A.1); top-left is best. Filtered QwenPRM adds an LLM-as-a-judge filtering stage to the same rollouts, raising F1 but discarding half of the labeled samples ($155\times$).

Answer diversity explains the IG-RAWR gap. To test whether answer diversity drives the IG-**RAWR** gap, we stratify problems by the number of distinct correct formulations among the K candidates (Figure 2, right). With 1–2 formulations (typical of math), IG scores 97.4% vs. 91.8% for **RAWR**. At 22–32 formulations, IG falls to 55.2% while **RAWR** stays at 93.2%, confirming that single-reference scoring breaks down when many equivalent answers exist and the $\max$ operator recovers the signal.

Computational efficiency. **RAWR** needs a single teacher-forced scoring pass per trace with KV-cache reuse and *generates nothing during labeling*: it scores the traces already sampled for training, so it adds none of the expensive output tokens rollouts produce. With output priced at $5\times$ input, it is 23–$77\times$ cheaper than rollout-based labeling, and matches unfiltered QwenPRM and exceeds MATHSHEPHERD on `ProcessBench` Step F1 at $77\times$ lower cost (Figure 3, Section A.1). This allows larger training sets for the same budget, and downstream PRM performance improves consistently with dataset size (Appendix C).

5.3 PRM PERFORMANCE

Baselines. We compare against methods spanning the full supervision spectrum (Table 2): no reward model (single sampling, majority voting); outcome-level supervision; implicit step signals derived from outcome training (ImplicitPRM (Yuan et al., 2024), OVM (Yu et al., 2024)); and explicit process supervision (MATHSHEPHERD (Wang et al., 2024a), Granite PRM (IBM Granite Alignment Team, 2026), and QwenPRM 7B (Zhang et al., 2025)). Granite PRM uses OMEGAPRM's labeling method, but on responses from larger generators and with extra math and code fine-tuning. Because these baselines use different backbones and training sets, we additionally train controlled variants

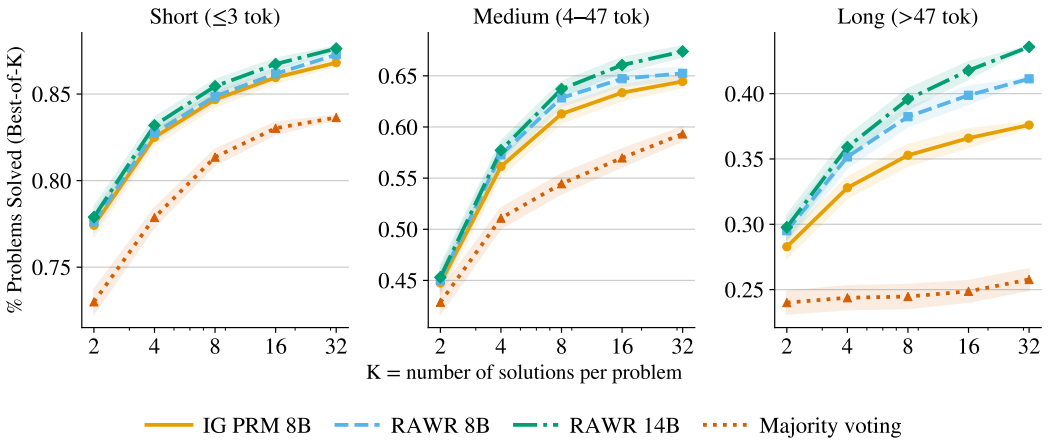

Figure 4: Best-of-K accuracy as a function of K, stratified by ground-truth answer length, for PRMs trained on IG labels and on **RAWR** labels. The **RAWR**-IG gap is negligible for short answers and widens as answers grow longer, confirming that single-reference IG degrades on structured, long-form outputs while **RAWR** remains robust. Shaded bands show ± 1 standard deviation across repeated subsamples of K answers per problem.

Table 2: Best-of-K accuracy (K=32) across eight benchmarks. Bold indicates the best result and underline the second best, computed separately within the 8B group and globally for **RAWR** 14B. Shaded rows are our methods; **RAWR** 8B achieves the highest average performance among 8B models.

Method	MATH	GSM	PubMed	HumanEval	BigCode	BIRD	AIME	Average
Single sampling	44.5%	73.3%	36.2%	55.5%	23.3%	19.6%	17.4%	38.5%
Majority voting	67.3%	91.2%	46.8%	66.1%	27.6%	37.6%	29.0%	52.2%
OVM	62.8%	92.0%	59.4%	76.3%	34.9%	45.1%	26.1%	56.6%
MathShepherd	56.1%	89.5%	49.7%	52.5%	29.5%	35.9%	23.0%	48.0%
ImplicitPRM	59.3%	89.9%	44.3%	77.8%	29.3%	35.6%	19.6%	50.8%
Granite PRM	62.5%	90.9%	49.8%	61.6%	31.4%	35.4%	27.0%	51.2%
QwenPRM 7B	74.3%	**94.0%**	54.1%	69.7%	31.5%	36.8%	**37.0%**	56.8%
IG 8B (ours)	73.1%	93.3%	49.0%	84.5%	43.0%	44.2%	35.0%	60.3%
ORM 8B (ours)	73.8%	93.7%	**62.0%**	84.0%	44.1%	**47.8%**	29.0%	62.1%
RAWR 8B (ours)	**74.8%**	93.8%	57.7%	**84.6%**	**46.8%**	46.2%	32.5%	**62.3%**
RAWR 14B (ours)	**75.0%**	**94.0%**	59.9%	**85.6%**	**47.1%**	**50.1%**	32.0%	**63.4%**

(**RAWR** 8B, **RAWR** 14B, IG 8B, ORM 8B) that share the Ministral 3 backbone and training data and differ only in the labeling signal (and size, for the 14B variant).

Results. **RAWR** 8B achieves 62.3% average accuracy, the best among all baselines, and scaling to 14B (**RAWR** 14B) adds +1.1pp. This confirms the prediction from Section 3.2: the IG-**RAWR** gap is small on short-answer mathematics (+1.7pp on MATH, +0.5pp on GSM8K) but large on code and SQL with long, compositional answers (+3.8pp on BigCodeBench, +2.0pp on BIRD), while QwenPRM 7B is at its strongest on mathematics, its training domain, but lags on code and SQL. Figure 4 makes the length dependence explicit across all benchmarks: the IG-**RAWR** gap is negligible for short answers (≤ 3 tokens) and largest for long ones (> 47 tokens), exactly where many equivalent formulations exist. A paired bootstrap over test problems (10K resamples, common problems) confirms **RAWR** significantly outperforms QwenPRM 7B ($p < 0.001$) and IG 8B (p=0.025); as **RAWR** and IG differ only in the labeling signal, this isolates the contrastive term. The gap to our ORM baseline is within noise (p=0.73), an expected tie: the ORM is trained on the verifier's perfect final-answer labels, which best-of-K scores directly, making it an upper bound on that task.

5.4 Out of Distribution Evaluation

Domain Shift. We evaluate on the AtomicPhysics subset of UGPhysics (Xu et al., 2025), a physics domain absent from our training set. **RAWR** 8B achieves 14.7% best-of-K accuracy, best

among 8B methods, and **RAWR** 14B reaches 15.1%, nearly doubling single-sampling performance (Table 3).

Chain-of-thought generator shift. `ProcessBench` (Zheng et al., 2025) evaluates step-level error localization in reasoning traces using Step F1 as its official metric. We additionally introduce Trace F1, which measures whether a trace is correctly classified as containing an error. These traces were generated by models unseen during training (Qwen, Llama). As reported in Table 3, **RAWR** 8B achieves 73.2 Trace F1, close to QwenPRM 7B (75.0) and substantially outperforming MathShepherd (41.1) and Granite PRM (51.9). On Step F1, our models score lower (39.8 for **RAWR** 8B vs. 73.1 for QwenPRM 7B), indicating that **RAWR** labels carry more noise at the individual step level. In contrast, QwenPRM adds a separate LLM-as-a-judge consensus filtering stage on top of its rollouts. It reduces this noise but discards half of the labeled samples, doubling the labeling cost to $155\times$ that of **RAWR** (Section A.1); without it, their ProcessBench Step F1 drops to 40.2 at $77\times$ our cost, the same as MATHSHEPHERD. Despite noisier step-level labels, **RAWR** 8B exceeds all non-QwenPRM baselines on Trace F1 (see Appendix E).

Table 3: External evaluation. `UGPhysics`: best-of-K accuracy (OOD domain). `ProcessBench` Trace: F1 for classifying whether a trace contains an error (introduced by us). `ProcessBench` Step: official F1 for localizing the first incorrect step. Bold indicates the best result and underline the second best, computed separately within the 8B group and globally for **RAWR** 14B.

Method	`UGPhysics`	`ProcessBench` Trace	`ProcessBench` Step
Single sampling	8.0%	—	—
Majority voting	11.5%	—	—
OVM	9.2%	33.3	18.6
MathShepherd	10.0%	41.1	31.5
ImplicitPRM	10.6%	68.6	21.2
Granite PRM	11.0%	51.9	<u>51.4</u>
QwenPRM 7B	13.4%	**75.0**	**73.1**
IG 8B (ours)	<u>13.7%</u>	70.1	35.5
ORM 8B (ours)	10.4%	72.8	—
RAWR 8B (ours)	**14.7%**	<u>73.2</u>	39.8
RAWR 14B (ours)	**15.1%**	**75.1**	40.8

6 CONCLUSION

RAWR replaces single-reference IG with a contrastive log-likelihood signal: how much a step widens the gap between the best-supported correct and incorrect answers. It matches IG on mathematics and pulls ahead on code and SQL, where many equivalent formulations exist (Figure 2), while generalizing well out of distribution (Table 2, Table 3). Its $\mathcal{O}(N)$ labeling complexity scales the method across four domains and cheaply into new ones.

Scope and limitations. **RAWR** makes three deliberate tradeoffs. First, it needs a binary validator, so it applies only to verifiable domains (mathematics, code, SQL), not open-ended tasks such as dialogue; where the validator is imperfect, label quality degrades gracefully, sub-linearly in its error rate (Appendix D). Second, it trades some per-step precision for throughput: without consensus filtering its step labels are noisier than QwenPRM's (the Step F1 gap, Table 3), whose filtered labels cost $155\times$ more to produce; the filtering stage can still be applied on top of **RAWR** labels when needed. The noise is absorbed at trace level, where our PRMs match or exceed all baselines on Trace F1 and best-of-K. Third, step-level validation against human annotations is possible only in mathematics, where step labels are available.

Outlook. Two extensions follow. Producing per-step scores without rollouts, **RAWR** can serve as a dense reward in RL pipelines at the cost of a forward pass; and since its only domain-specific component is the validator, it extends to multimodal and agentic settings where validators already exist, reducing the barrier to step-level supervision from months of expert annotation to a single scoring pass.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2357–2367, 2019.

Anthropic. Claude API pricing. `https://docs.anthropic.com/en/docs/about-claude/pricing`, 2026. Accessed 26 September 2026.

Hanjie Chen, Faeze Brahman, Xiang Ren, Yangfeng Ji, Yejin Choi, and Swabha Swayamdipta. REV: Information-theoretic evaluation of free-text rationales. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Google. Gemini developer API pricing. `https://ai.google.dev/gemini-api/docs/pricing`, 2026. Accessed 26 September 2026.

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

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *NeurIPS*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021b.

Hugging Face. Math verify: A verification library for mathematical reasoning, 2024. URL `https://github.com/huggingface/Math-Verify`.

IBM Granite Alignment Team. Granite-3.3-8B-Math-PRM-v2. `https://huggingface.co/ibm-granite/granite-3.3-8b-math-prm-v2`, 2026. Model card.

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

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, et al. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240*, 2025.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems*, 36, 2024.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Making slow thinking faster: Compressing llm chain-of-thought via step entropy. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

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

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. *arXiv preprint arXiv:2204.05660*, 2022.

OpenAI. OpenAI API pricing. `https://platform.openai.com/docs/pricing`, 2026. Accessed 26 September 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Corentin Royer, Bjoern Menze, and Anjany Sekuboyina. Multimedeval: A benchmark and a toolkit for evaluating medical vision-language models. *arXiv preprint arXiv:2402.09262*, 2024.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

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

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. Understanding chain-of-thought in llms through information theory. *arXiv preprint arXiv:2411.11984*, 2024.

Tim Tully, Joff Redfern, Deedy Das, and Derek Xiao. 2025: The state of generative AI in the enterprise. Menlo Ventures report, `https://menlovc.com/perspective/2025-the-state-of-generative-ai-in-the-enterprise/`, 2025. December 9, 2025.

P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *ACL*, pp. 9426 – 9439, 2024a. doi: 10.18653/v1/2024.acl-long.510.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. In *EMNLP (Findings)*, pp. 7309–7319. Association for Computational Linguistics, 2024b.

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

Xin Xu, Qiyun Xu, Tong Xiao, Tianhao Chen, Yuchen Yan, Jiaxin Zhang, Shizhe Diao, Can Yang, and Yang Wang. Ugphysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. *arXiv preprint arXiv:2502.00334*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36:11809–11822, 2023.

Fei Yu, Anningzhe Gao, and Benyou Wang. OVM, outcome-supervised value models for planning in mathematical reasoning. In *NAACL-HLT (Findings)*, pp. 858–875. Association for Computational Linguistics, 2024.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

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

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)*, pp. 1009–1024, 2025.

Danna Zheng, Mirella Lapata, and Jeff Pan. Archer: A human-labeled text-to-SQL dataset with arithmetic, commonsense and hypothetical reasoning. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 94–111, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.6. URL https://aclanthology.org/2024.eacl-long.6/.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

A Complexity Derivation

We analyze complexity in terms of *tokens processed* by each method. Let N be the number of CoT steps to label, $\bar{s}$ the average tokens per step, T the average number of tokens in an answer y, and S the number of sampled candidate answers used in the Monte Carlo approximation. Denote by $|q|$ the number of tokens in the question. For comparison, let M be the number of rollouts per prefix used by other methods. We count tokens processed by the language model (generation or scoring). We also assume standard KV-caching is available so that once a prefix is processed we do not reprocess its earlier tokens when scoring different completions.

MathShepherd (naive rollouts). At each step i (for $i = 1, \ldots, N$) the naive method generates M rollouts continuing the prefix $R_{1:i}$. Each rollout has expected length approximately $(N - i)\bar{s}$ tokens. Therefore the total expected number of rollout tokens is

$$\mathbb{E}[\text{Tokens}_{\text{MS}}] \;=\; M\bar{s}\sum_{i=1}^{N}(N-i) \;=\; M\bar{s}\cdot\frac{N(N-1)}{2}.$$

Since $\frac{N(N-1)}{2} = \mathcal{O}(N^2)$, this method costs $\mathcal{O}(M\bar{s}N^2)$ tokens (i.e., $O(N^2)$ in N).

OmegaPRM (binary-search on first-failure). OmegaPRM reduces the number of *checked prefixes* from N to $\lceil \log_2 N \rceil$: only $\mathcal{O}(\log N)$ indices are queried by binary search. At each query, M rollouts are generated to the point of conclusion. If the location of the first incorrect step is assumed to be uniformly distributed over the N steps of the CoT, then the *expected* rollout length is $N/2$ steps.

Therefore, the expected number of tokens processed is

$$\mathbb{E}[\text{Tokens}_{\Omega}] \;=\; M\bar{s}\,\tfrac{N}{2}\,\lceil \log_2 N \rceil$$

Thus, the expected asymptotic complexity of OmegaPRM is $\mathcal{O}(N \log N)$ tokens. This improves over the quadratic cost of MathShepherd, though still grows faster in N than the linear complexity of our method.

Our method: RAWR To compute NetInfo$_i$ for all $i = 1 \ldots N$ we must evaluate $\mathcal{I}_i(y)$ for every sampled candidate $y \in \mathcal{Y}$. With KV-caching the natural implementation is:

1. Process the prompt q and the entire target CoT $r_{1:N}$ once, storing the KV-cache and the boundary of each step i. This costs $|q| + \sum_{i=1}^{N} |r_i| \approx |q| + N\bar{s}$ input tokens.
2. For each step i and each sampled answer y compute the token log-probabilities of y conditioned on $q, r_{1:i}$. With KV caching this requires only processing the answer tokens, i.e. about T tokens per (prefix,answer) pair.

Thus the scoring tokens are approximately $N \times S \times T$. Adding the one-time prefix pass and the baseline $i = 0$ scores (which cost $\approx ST$) gives:

$$\mathbb{E}[\text{Tokens}_{\textbf{RAWR}}] \;=\; |q| + N\bar{s} + (N+1)\,S\,T$$

If S and T are treated as constants independent of N, the asymptotic complexity is in $\mathcal{O}(N)$. Thus **asymptotically in N and for constant S, T the RAWR method has a linear complexity,** whereas MathShepherd has a quadratic complexity and OmegaPRM a complexity in $\mathcal{O}(N \log N)$.

A.1 Empirical token count estimation

The token counts reported in Table 5 are estimated by instantiating the formulas below on the `PRM800K` dataset. For each trace we measure the number of reasoning steps N, the average tokens per step $\bar{s}$, the number of tokens in the question $|q|$, and the total tokens across the distinct candidate answers sampled for its question, T_{alt}. We set M=8 rollouts per prefix (following the original papers) and sum over all traces. All methods receive the same KV-cache assumption as the derivation above: a trace is prefilled once and its cache is branched at every step boundary, so a rollout or a scored answer only pays for its own new tokens.

Input vs. output tokens. A raw token count treats every token equally, but the two kinds of tokens a method processes have very different cost. *Input* (prefill) tokens are read by the model in a single parallel pass; *output* (decode) tokens are generated one at a time and are far more expensive, both in wall-clock throughput (decode is memory-bandwidth bound) and in price. Table 4 lists the current API prices of Anthropic, OpenAI and Google, which together account for 88% of enterprise LLM API usage (Tully et al., 2025): every listed model bills output at 5–8.3× the input rate, with a median of 5× (Anthropic, 2026; OpenAI, 2026; Google, 2026). We therefore price output at 5× input for the cost-weighted totals. This distinction is decisive here, because the rollout-based methods spend almost their entire budget *generating* rollouts (output), whereas **RAWR** performs only teacher-forced scoring—every token is read, none is generated, so its cost is 100% input. Table 5 reports the two separately. The conclusion does not depend on the multiplier: counting raw tokens (1×), rollout-based labeling still costs 4.8–32× more than **RAWR**.

Table 4: API list prices (USD per million tokens; standard tier, short context) of the three providers that together account for 88% of enterprise LLM API usage (Tully et al., 2025), accessed 26 September 2026 (Anthropic, 2026; OpenAI, 2026; Google, 2026). Every model bills output at 5–8.3× the input rate (median 5×), the multiplier used in Table 5.

Provider (API share)	Model	Input	Output	Output / input
Anthropic (40%)	Claude Fable 5.1	10.00	50.00	5.0×
	Claude Opus 5.5	4.00	20.00	5.0×
	Claude Sonnet 5	2.00	10.00	5.0×
	Claude Haiku 4.5	1.00	5.00	5.0×
OpenAI (27%)	GPT-6 Astra	10.00	50.00	5.0×
	GPT-6 Sol	2.00	10.00	5.0×
	GPT-6 Luna	0.10	0.50	5.0×
Google (21%)	Gemini 3.1 Pro (preview)	2.00	12.00	6.0×
	Gemini 3.8 Flash	0.75	3.75	5.0×
	Gemini 3.5 Flash-Lite	0.30	2.50	8.3×

Table 5: Tokens processed by each labeling method to produce the same number of labeled samples on PRM800K, split into input (prefill/scoring) and output (generation), after normalizing by each method's retention rate ρ. CoT generation is shared across all methods and excluded. Rollout-based methods spend 94–98% of their tokens on output; **RAWR** generates none. The cost-weighted total prices output at 5× input, the median list-price ratio of the major API providers (Table 4); the last two rows give the cost relative to **RAWR** at that rate and, for reference, counting raw tokens (1×). QWENPRM uses the MATHSHEPHERD rollouts plus a separate LLM-as-a-judge filtering stage; its judge verdict tokens are excluded, so its cost is a lower bound. Granite PRM uses Automatic Process Supervision (the OMEGAPRM MCTS labeling method), so we estimate its labeling cost as equal to OMEGAPRM's; this counts the labeling stage only, not its additional fine-tuning.

	Qwen PRM	MathShepherd	OmegaPRM	Granite PRM	**RAWR**
Input tokens	1.8×10^8	0.44×10^8	0.44×10^8	0.44×10^8	$\mathbf{1.5 \times 10^8}$
Output tokens	45.8×10^8	22.9×10^8	6.6×10^8	6.6×10^8	**0**
Cost-weighted (out = 5×)	230.6×10^8	114.9×10^8	33.6×10^8	33.6×10^8	$\mathbf{1.5 \times 10^8}$
relative to **RAWR**	155×	77×	23×	23×	**1×**
relative to **RAWR**, raw tokens (out = 1×)	32×	15.7×	4.8×	4.8×	**1×**

Because each method discards a different fraction of samples before PRM training (e.g., samples where all rollouts are correct or all are incorrect), the raw token counts do not reflect the cost of reaching the same final dataset size. To ensure a fair comparison, we normalize every method by its *retention rate* ρ—the fraction of labeled samples that survive filtering—so that the reported count reflects the total tokens that must be processed to produce the same number of usable training examples:

$$\text{Tokens}_{\text{adjusted}} = \frac{\text{Tokens}_{\text{raw}}}{\rho}. \tag{9}$$

MATHSHEPHERD and OMEGAPRM report no filtering stage (ρ=1), QWENPRM retains approximately 50% of samples after consensus filtering ($\rho_{\text{Qwen}} = 0.50$), and **RAWR** retains approximately 83% ($\rho_{\textbf{RAWR}} = 0.83$).

The per-trace formulas, split into input and output tokens, are:

MATHSHEPHERD. The trace is read once; from each prefix $R_{1:i}$, M rollouts continue for the remaining $(N - i)$ steps:

$$\text{In}_{\text{MS}} = |q| + N\bar{s}, \qquad \text{Out}_{\text{MS}} = M\bar{s}\,\frac{N(N-1)}{2}.$$

OMEGAPRM. Binary search checks $\lceil \log_2 N \rceil$ prefixes, each with M rollouts of expected length $N/2$ steps:

$$\text{In}_{\Omega} = |q| + N\bar{s}, \qquad \text{Out}_{\Omega} = M\bar{s}\,\frac{N}{2}\,\lceil \log_2 N \rceil.$$

QWENPRM. The MATHSHEPHERD rollouts, plus a separate LLM-as-a-judge filtering stage that reads the question and the full trace:

$$\text{In}_{\text{Qwen}} = \underbrace{|q| + N\bar{s}}_{\text{rollouts}} + \underbrace{|q| + N\bar{s}}_{\text{judge}}, \qquad \text{Out}_{\text{Qwen}} = M\bar{s}\,\frac{N(N-1)}{2}.$$

The judge's own verdict tokens are unknown and excluded, so this is a *lower bound* on the filtering cost. Most of the filtering cost comes from the retention rate: discarding half of the labeled samples doubles the tokens needed per usable sample.

RAWR. A single pass over the trace, plus scoring every candidate answer after each prefix (including the baseline i=0); nothing is generated:

$$\text{In}_{\textbf{RAWR}} = |q| + N\bar{s} + (N+1)\,T_{\text{alt}}, \qquad \text{Out}_{\textbf{RAWR}} = 0.$$

B TRAINING SET DETAILS

To construct a diverse and comprehensive training corpus, we aggregate a wide range of publicly available benchmarks spanning mathematical reasoning, scientific question answering, program synthesis, and text-to-SQL semantic parsing. Table 6 summarizes the datasets used, along with the number of unique problems and total candidate answers derived from each source.

For some coding benchmarks, we adapt a subset of the available samples for training to preserve held-out data for evaluation. Specifically, we use 64 samples from `HumanEval` and 640 samples from `BigCodeBench` as training data, reserving the remainder for testing. For `SynSQL`, we subsample 20,000 problems from the "highly complex" subset of the dataset for training.

Overall, the combined training mixture spans **232K problems** and, after generation and filtering, results in **1.49M candidate answers**, providing broad coverage of reasoning tasks and ensuring varied supervision across mathematical, scientific, and programming domains. Table 7 reports statistics of the generated CoTs, and Figure 5 shows the distribution of CoT lengths.

Table 6: Training dataset composition.

Dataset	# Problems	# Candidate answers
`MATH500` (Lightman et al., 2023)	12,000	85,248
`GSM8K` (Cobbe et al., 2021)	8,790	31,768
`MathQA` (Amini et al., 2019)	29,837	204,974
`AquaRat` (Ling et al., 2017)	97,700	589,446
`NumGLUE` (Mishra et al., 2022)	41,018	265,532
`PubMedQA` (Jin et al., 2019)	1,000	3,727
`HumanEval` (Chen et al., 2021)	64	480
`BigCodeBench` (Zhuo et al., 2024)	640	5,080
`APPS` (Hendrycks et al., 2021a)	5,000	26,768
`BIRD` (Li et al., 2024)	9,428	69,760
`Spider` (Yu et al., 2018)	7,000	46,233
`ARCHER` (Zheng et al., 2024)	414	3,304
`SynSQL` (Li et al., 2025)	20,000	159,104
Total	**232,891**	**1,491,424**

Table 7: Statistics of the generated CoTs. The BoK columns show the oracle best-of-K: the percentage of problems where at least one of the K samples is correct, whereas the % correct columns show the percentage of generations that are correct.

Dataset	Avg # step	Train BoK	Train % correct	Test BoK	Test % correct
`MATH500`	8.50	86.57%	33.37%	88.98%	51.10%
`GSM8K`	7.52	98.55%	75.81%	97.72%	84.30%
`MathQA`	8.08	81.22%	38.52%	-	-
`AquaRat`	8.29	71.65%	28.19%	-	-
`NumGLUE`	5.80	86.05%	38.34%	-	-
`PubMedQA`	10.08	87.35%	24.58%	86.97%	44.25%
`HumanEval`	7.28	98.44%	31.64%	95.96%	60.01%
`BigCodeBench`	8.07	69.12%	11.82%	68.07%	27.04%
`APPS`	8.01	51.41%	12.71%	-	-
`BIRD`	7.25	67.11%	17.71%	74.49%	34.81%
`Spider`	6.46	90.07%	44.81%	-	-
`ARCHER`	7.69	42.86%	6.31%	-	-
`SynSQL`	7.48	36.54%	6.70%	-	-
`UGPhysics`	7.92	-	-	26.99%	9.11%
`AIME`	11.57	-	-	54.00%	18.70%

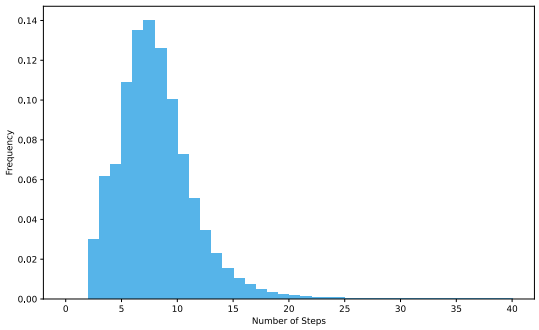

Figure 5: Distribution of the number of steps in the CoTs of all the datasets combined.

C Dataset Scaling

We investigate how the scale of the PRM training dataset affects model performance. Using the Ministral 3 8B backbone, we train PRMs on four dataset sizes: 1.2M, 500K, 250K, and 100K samples. We then evaluate each resulting model on the full suite of best-of-K tasks and report the average. As shown in Figure 6, performance improves consistently with increased data scale, indicating that larger preference-label datasets provide more robust learning signals for reward modeling. These results highlight the value of **RAWR** as it allows for easier scaling up to millions of samples.

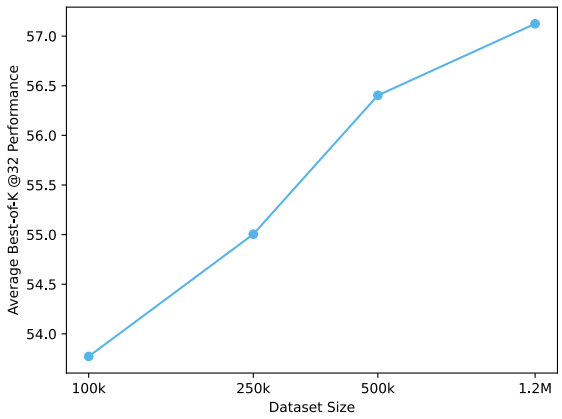

Figure 6: Average best-of-K performance over the evaluation tasks for PRMs trained on different subsets of the training set, ranging from 100K to the full 1.2M samples.

D RAWR ABLATIONS

We ablate two design choices: the aggregation rule over alternative answers ($\max$ vs. sum) and the baseline anchor. **RAWR** anchors each step against the pre-reasoning baseline NetInfo_0 (Equation 7); an alternative is the strictly local increment $\text{NetInfo}_i - \text{NetInfo}_{i-1}$, which measures only the change contributed by step r_i. Table 8 reports balanced accuracy for all four combinations, micro-averaged across the 13 training datasets. $\max$ outperforms sum under both anchors (by 0.4pp with the step-0 anchor), and anchoring at step 0 is uniformly better (e.g. 82.0% vs. 80.5% for $\max$). Reasoning traces frequently drift into error gradually, with no single step dropping the answer likelihood sharply; the successive difference cancels this accumulated drift along with the prior, whereas anchoring at step 0 retains it. This is why we keep the step-0 anchor rather than reformulate **RAWR** as a strictly marginal signal.

Table 8: Balanced accuracy (%) of **RAWR** labels under each aggregation rule ($\max$, sum) and baseline anchor, micro-averaged across the 13 training datasets. $\max$ outperforms sum under both anchors, and anchoring at the pre-reasoning baseline (NetInfo_0) outperforms the strictly local successive difference (NetInfo_{i-1}) under every aggregation rule.

Aggregation	Baseline anchor (r_0)	Successive diff. (r_{i-1})
`max`	**82.01**	80.46
`sum`	81.60	80.09

Likelihood evolution along the trace. **RAWR** rests on the assumption that a correct reasoning prefix raises the model's likelihood of the gold answer while a flawed one lowers it. We test this directly on `ProcessBench` (`GSM8K` and `MATH`, 1,241 traces), the shortest-answer datasets available (1-3 answer tokens) and thus the strongest case for a saturated prior. Under teacher forcing we measure the length-normalized probability of the gold answer $p_i(y^\star)$ given the prefix $R_{1:i}$, binned by normalized step index i/N (Table 9). Before any reasoning ($i/N = 0$) the two groups are indistinguishable (0.149 vs. 0.140); as reasoning accumulates they separate monotonically in opposite directions, ending at 0.995 for correct traces and 0.027 for incorrect ones—a 0.97 gap. This is the behaviour the signal assumes, in the regime least favourable to it.

Table 9: Median length-normalized gold-answer probability $p_i(y^\star)$ against the normalized step index i/N, teacher-forced on 1,241 `ProcessBench` traces. Correct and incorrect traces are indistinguishable before reasoning and diverge monotonically as it accumulates.

i/N	0.0	0.2	0.4	0.6	0.8	1.0
Correct (n=484)	0.149	0.153	0.165	0.376	0.886	0.995
Incorrect (n=757)	0.140	0.133	0.113	0.107	0.069	0.027

Prior saturation and calibration. A likelihood-based signal could lose its range if the model is already confident in the answer before any reasoning. We find this regime is rare and does not hurt label quality. First, saturation is rare even in the shortest-answer regime: the median pre-reasoning prior p_0 is 0.15 on `ProcessBench`, no `GSM8K` problem and only 1.4% of `MATH` problems exceed $p_0 > 0.9$, and the same holds at scale (1.5% of our full 85,248-trace `MATH` set). Second, where it does occur the labels hold: on the saturated subset, balanced accuracy is $89.9\% \pm 0.8$, above the $83.0\% \pm 0.1$ over the full set (Table 10). **RAWR** thresholds a *relative* difference, so it requires only that the score order correct above flawed reasoning, not that any probability be calibrated; a badly calibrated model can still order steps correctly.

Sensitivity to verifier noise. **RAWR** depends on a binary validator to partition candidates. Our validators are execution- or parser-based and near-ground-truth, but we quantify sensitivity to an imperfect one by corrupting the partition: at noise level ε we independently flip each candidate's label with probability ε and recompute **RAWR** labels under the corrupted partition. Table 11 reports balanced accuracy on `BIRD` and `MATH` over 7 seeds per level. Degradation is sub-linear in ε: a 5%

Table 10: Balanced accuracy of **RAWR** labels on the full MATH training set versus the high-prior subset ($p_0 > 0.9$), same protocol as the main paper. Spreads are ± 1 std over a bootstrap. Labels on saturated priors are more accurate than on the full set.

	Balanced acc.	Traces
Full set	$83.0\% \pm 0.1$	85,248
High-prior ($p_0 > 0.9$)	$89.9\% \pm 0.8$	1,298

error rate costs 3.1 points on BIRD and 2.3 on MATH, and even at 10% the labels retain 80.2% and 76.9%. This follows from the $\max$ aggregation—a flipped label moves the signal only if it becomes the argmax of the set it is wrongly placed in, so a misclassified low-probability answer is absorbed with no effect.

Table 11: Balanced accuracy of **RAWR** labels under a verifier that mislabels each candidate with probability ε (7 seeds per level). Degradation is sub-linear and the labels remain usable across the full range tested.

Verifier noise ε	BIRD	Δ vs. clean	MATH	Δ vs. clean
0.00 (clean)	85.34%	—	81.59%	—
0.01	84.41%	−0.93	81.50%	−0.09
0.05	82.22%	−3.13	79.30%	−2.29
0.10	80.15%	−5.19	76.94%	−4.65

Sampling budget K. **RAWR** approximates the correct and incorrect answer spaces from K sampled candidates. We ablate this budget on 100 MATH problems, drawing a 512-candidate reference pool and measuring downstream balanced accuracy when only s candidates are subsampled (Table 12). Accuracy rises steeply and plateaus by $s \approx$ 20-30, staying within 4 points of the full-pool reference. Our K=32 therefore sits on the plateau, and the method degrades gracefully below it (s=10, under a third of the budget, stays within 5.2 points of the reference). This follows from the low sample complexity of the $\max$ aggregator, which needs only one high-probability correct formulation to appear rather than full coverage of the equivalence class.

Table 12: Downstream balanced accuracy as a function of the number of subsampled candidates s (100 MATH problems, 512-candidate reference pool). Accuracy plateaus by $s \approx$ 20-30; the paper's K=32 sits on the plateau.

s	5	10	20	30	40	512 (ref)
Balanced acc.	69.8%	72.7%	73.9%	74.2%	74.1%	77.9%
Gap to reference	8.1	5.2	4.0	3.7	3.8	—

Finally, we demonstrate the higher performance of **RAWR** compared to IG in Figure 7. This shows the two signals for the same sample. The CoT reaches a correct answer but not the gold one from the dataset. As we approach the last step, the likelihood of generating the answer that IG expects decreases whereas **RAWR** still performs well.

Example of a *correct* CoT for the Bird dataset (database schema omitted)

Question: Indicate the title of all podcasts in the `fiction` category.

1. Identify podcasts labeled as `fiction`.
2. The `podcasts` and `categories` tables contain podcast metadata and category assignments, respectively.
3. Join the tables on `podcast_id` to combine podcast information with category information.
4. Filter rows where the category equals `'fiction'`.
5. Select unique podcast titles. The resulting SQL query is:

```
SELECT DISTINCT podcasts.title
FROM podcasts
INNER JOIN categories
    ON podcasts.podcast_id = categories.podcast_id
WHERE categories.category = 'fiction';
```

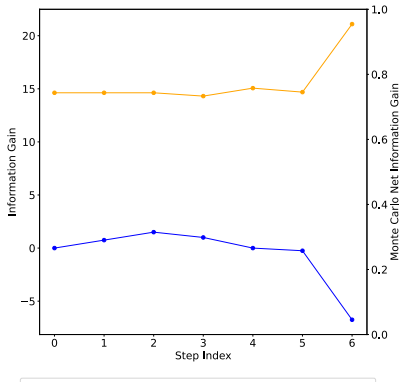

Figure 7: Example of a sample that is misclassified under IG but correctly identified under **RAWR**, illustrating the sharper discriminative behavior of the max-based information signal. The right axis shows the NetIG score of **RAWR**.

E PROCESSBENCH ERROR LOCALIZATION ANALYSIS

To better understand the Step F1 gap between **RAWR** and QwenPRM on `ProcessBench`, we analyze the distribution of *error step offsets*: the difference between the predicted error step and the ground-truth error step. An offset of 0 indicates exact localization; negative offsets mean the PRM flags an error earlier than ground truth, and positive offsets mean it flags it later.

Figure 8 (left) shows the offset distribution for QwenPRM 7B. The distribution is tightly concentrated at offset 0 ($\sim$158 exact matches), with minimal spread, consistent with its high Step F1 of 73.1. Figure 8 (right) shows the same analysis for **RAWR** 8B. The mode is still at offset 0 ($\sim$107 exact matches), but the distribution is wider, with notable mass at offsets ± 1 and ± 2.

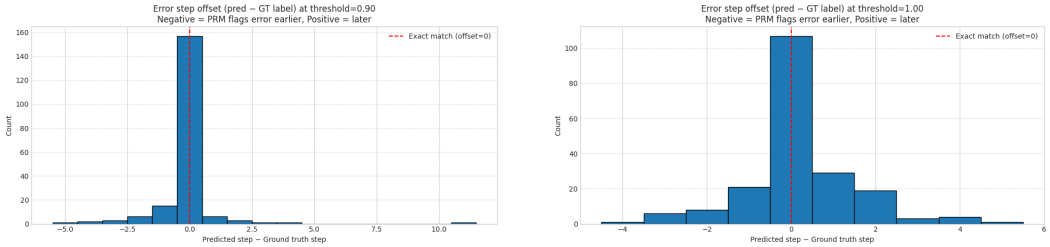

Figure 8: Error step offset distributions on `ProcessBench`. **Left:** QwenPRM 7B, tightly concentrated at offset 0, indicating precise step-level localization. **Right:** **RAWR** 8B, with the mode still at offset 0 but wider spread predominantly in $[-2, +2]$. The right skew (about 56 traces at positive offsets vs. 36 at negative ones) indicates a tendency to flag errors later than ground truth.

A further check distinguishes **RAWR** from an outcome-level signal, which would concentrate its flags at the final step of a trace. Table 13 reports, among incorrect `ProcessBench` solutions that **RAWR** flags, the fraction whose first flagged step falls *before* the last step. Across all four subsets this exceeds 77%, confirming that the signal localizes errors inside the reasoning rather than defaulting to the outcome.

Table 13: Percentage of flagged incorrect `ProcessBench` solutions whose first flagged step precedes the final step. An outcome-level signal would place its flag at the last step, giving values near 0.

GSM8K	MATH	Olympiad	OmniMath
82.8%	77.2%	81.5%	80.2%

F SCORING PROMPT TEMPLATE

RAWR scores each candidate answer by teacher-forcing it after a reasoning prefix. The generator and the scoring pass share the prompt below: a system instruction to reason in `[STEP]`-delimited steps and to close with the answer inside `<final-answer>` tags, four few-shot exemplars in that format, then the target question. To score a prefix, we truncate the trace after step i, append the candidate answer, and compute $\mathcal{I}_i(y)$ on the answer tokens only (those following `\boxed{` for mathematics, or inside `<final-answer>` for the other domains). The exemplars are drawn from the corresponding training domain; we show the mathematics template as a representative instance.

RAWR scoring prompt (mathematics; exemplar bodies abbreviated)

System. You are a careful reasoner. Solve the problem one step at a time. Begin each step with the marker `[STEP]`. After the final step, give the final answer between `<final-answer>` and `</final-answer>` tags, with the numeric result in `\boxed{}`.
Few-shot exemplars ($\times 4$, one shown):
Question: ...
`[STEP]` ... `[STEP]` ...
`<final-answer>\boxed{...}</final-answer>`
Target.
Question: {question}