

SHARP: Span-level Hallucination Annotation for Reasoning Paths in Generated Mathematical Solutions

Timofey Garaev¹, Makar Gorokhovik¹, Alexey Korotkov¹, Maksim Savkin¹,
Iurii Vizilter¹, Andrey Savchenko^{2,3}

¹Moscow Independent Research Institute of Artificial Intelligence, ²SB AI Lab, ³HSE University

Correspondence: garaev.t@miriai.org

Abstract

We introduce SHARP, a compact, span-annotated dataset for hallucination detection in model-generated mathematical solutions. Unlike existing resources that rely on coarse step-level labels, SHARP provides fine-grained, span-level annotations generated through an automated pipeline. Our methodology synthesizes realistic reasoning traces with precise error markup, enabling the production of training data with minimal computational overhead. Despite being 10–30× smaller than existing open-source datasets, a 1.5B PRM fine-tuned on SHARP achieves state-of-the-art performance among 1.5B PRMs on ProcessBench.

1 Introduction

Modern large language models (LLMs) demonstrate strong capabilities on complex mathematical problems when generating step-by-step chain-of-thought solutions (Wei et al., 2022). However, these models remain prone to *hallucinations* (Maynez et al., 2020): plausible but logically or factually incorrect generated content. Even a single flawed logical step can propagate through a solution, ultimately corrupting the final answer. Detecting such errors is critical for improving the reliability of LLM-generated reasoning.

There are two common approaches for detecting hallucinations in reasoning traces: Outcome Reward Models (ORMs) and Process Reward Models (PRMs). ORMs assign a reward based on the final answer of a reasoning trajectory (Uesato et al., 2022). PRMs operate at the step level, assigning a reward to each intermediate step conditioned on the problem and all preceding steps (Zhang et al., 2025; Wang et al., 2024; He et al., 2024b; Cui et al., 2025; Xiong et al., 2024). Step-level supervision allows PRMs to identify hallucinated or incorrect steps within a reasoning chain. Because of this finer granularity, PRMs have become the

standard approach for step-level reasoning verification (Lightman et al., 2024). In this study, we focus exclusively on PRMs.

Current datasets for hallucination detection in reasoning suffer from two fundamental limitations. First, supervision is almost exclusively provided at the step level. In practice, hallucinations often occur at a finer granularity: arithmetic mistake or unjustified transformation embedded within an otherwise valid step. Second, many datasets are typically large (hundreds of thousands of examples), which makes their construction and PRMs training on them costly.

To address these limitations, we introduce **SHARP** (Span-level **H**allucination **A**nnotation for **R**easoning **P**aths), a dataset built by a fully automatic pipeline for generating span-level hallucination annotations in mathematical reasoning.

To the best of our knowledge, this is the first dataset designed specifically for span-level hallucination detection in mathematical chain-of-thought reasoning, generated fully automatically¹.

Our contributions can be summarized as follows:

- We propose a novel methodology for constructing a span-level hallucination detection dataset for mathematical reasoning, focusing on naturally occurring hallucinations rather than synthetically injected errors.
- We demonstrate that SHARP can be converted into effective step-level supervision for training PRMs. On ProcessBench (Zheng et al., 2025), our 1.5B model achieves state-of-the-art performance among 1.5B PRMs, and our 7B model reaches top-three performance among 7–8B PRMs while using less training data than the

¹Dataset: <https://huggingface.co/datasets/ZaandaTeika/SHARP>. Models: <https://huggingface.co/ZaandaTeika/Qwen2.5-Math-7B-SHARP-Span> and <https://huggingface.co/ZaandaTeika/Qwen2.5-Math-1.5B-SHARP-Span>; the SHARP-STEP checkpoints are available in the same namespace.

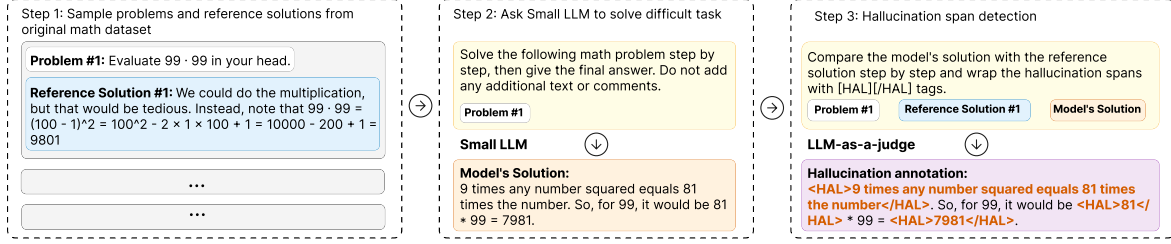


Figure 1: The proposed SHARP-Span annotation pipeline.

second-best model and trailing it by only 0.5 Avg. F1.

- Thorough ablation study shows that supervision derived from span-level annotations consistently outperforms step-level markup, suggesting that finer-grained hallucination annotations provide a more reliable basis for constructing step-level PRM training labels.

2 Related Work

The development of PRMs has primarily focused on scaling step-level supervision through human annotation or automated heuristics.

Human-Annotated Datasets. The leading resource in this domain, PRM800K (Lightman et al., 2024), contains approximately 800,000 human-labeled step-level annotations. Built upon the MATH dataset (Hendrycks et al., 2021), it uses GPT-4 (OpenAI, 2023) generated solutions. While highly accurate, the reliance on human experts makes such datasets difficult to scale. Examples of PRMs that use the PRM800K dataset for training include: ReasonEval-7B (Xia et al., 2025), RetrievalPRM-7B (Zhu et al., 2025), PathFinder-PRM-7B (Pala et al., 2025), and the R-PRM-7B SFT and DPO variants (She et al., 2025). Table 1 summarizes how SHARP compares to existing process-supervision datasets.

Automated Markup. To reduce the costs of human labeling, several automated frameworks have been proposed. Math-Shepherd (Wang et al., 2024) utilizes a Monte Carlo rollout strategy: an intermediate step is assigned a correctness score based on the proportion of completed paths that reach a correct final answer. While scalable, this approach introduces significant label noise; a valid step may lead to a wrong answer due to subsequent errors.

More recently, QwenPRM (Zhang et al., 2025) achieved state-of-the-art results by synthesizing

Dataset	Annotation	Size	Accessibility	Granularity
QwenPRM (Zhang et al., 2025)	Auto	1,500k	Closed	Step-level
RetrievalPRM (Zhu et al., 2025)	Auto	445k	Open	Step-level
R-PRM (She et al., 2025)	Auto	558k	Open	Step-level
PathFinder-PRM (Pala et al., 2025)	Auto	400k	Open	Step-level
EurusPRM-S1/S2 (Cui et al., 2025)	Auto	463k/493k	Open	Step-level
PRM800K (Lightman et al., 2024)	Manual	415k*	Open	Step-level
Math-Shepherd (Wang et al., 2024)	Auto	445k	Open	Step-level
RLHFlow-PRM (Xiong et al., 2024)	Auto	253k/273k	Open	Step-level
SHARP (ours)	Auto	19.1k	Open	Span-level

Table 1: Comparative analysis of SHARP against existing mathematical reasoning datasets. The columns represent: **Annotation**, the labeling method (Manual vs. Automatic); **Accessibility**, whether the data is publicly available. * - The size for PRM800K is reported after one-trajectory-per-sample conversion described in Appendix B.4; the effective size after filtering is 214,994 samples.

two strategies: Monte Carlo rollouts and model-based feedback from a strong external critic (Zheng et al., 2023). This hybrid approach produces a large-scale, low-noise dataset. However, like its predecessors, QwenPRM remains limited to step-level markup; its weights are open, but the underlying dataset is not.

Another example of a Qwen2.5-Math-based PRM is Skywork-o1-PRM (He et al., 2024b). The dataset used to train this model is not publicly available. In addition, RLHFlow has released popular PRMs (Xiong et al., 2024) based on Llama 3.1 (Grattafiori et al., 2024), whose authors adopt the Math-Shepherd approach to generate their training sets.

Benchmarks for Process Verification. The evaluation of these models has been standardized through benchmarks like **ProcessBench** (Zheng et al., 2025). The benchmark requires models to identify the first erroneous step across 3,400 test cases spanning various difficulty levels, including **GSM8K** (Cobbe et al., 2021), **MATH** (Hendrycks et al., 2021), **OlympiadBench** (He et al., 2024a), and **Omni-MATH** (Gao et al., 2024). While ProcessBench effectively evaluates step-level error detection, our dataset introduces a finer-grained su-

pervision signal through span-level hallucination annotations. This supervision can potentially support more precise localization of reasoning errors, although current evaluation remains step-level due to the absence of established benchmarks at span-level granularity.

3 Proposed SHARP Dataset

Our dataset construction follows a four-stage pipeline (Figure 1): problem selection, trajectory generation, span-level annotation and filtering procedure.

Problem Selection and Trajectory Generation.

We select GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) as the primary sources of problems, as they cover different levels of mathematical difficulty and span a broad range of mathematical domains. To generate reasoning trajectories, we use Mathstral-7B-v0.1 (Mistral AI, 2024), a math-specialized 7B language model with the prompt shown in Figure 4. Since the model is fine-tuned for mathematical reasoning, it produces coherent step-by-step solutions while still making naturally occurring reasoning errors. This makes its generations particularly suitable as raw material for hallucination detection: the resulting mistakes are less artificial and better reflect the types of failures that arise in mathematical chain-of-thought reasoning.

Span-Level Annotation. To convert raw reasoning traces into fine-grained supervision, we use gpt-oss-120B (OpenAI, 2025) as an automated annotator (Fig. 7). The model is prompted (Fig. 5) with a triplet consisting of the original problem statement, the ground-truth reference solution, and the generated reasoning trace. As output, we obtain an annotated chain of reasoning in which erroneous spans are explicitly marked with <HAL> tags.

Filtering Procedure. After annotation, we remove low-quality annotations. This filtering is motivated by the hypothesis that unreliable automatic markup often manifests itself through annotation artifacts, such as excessively long hallucination spans, an abnormally large number of marked fragments, or malformed generations. In the final pipeline, we use percentile-based filtering: malformed outputs are discarded, and examples with span statistics above the 95th percentile of the empirical distribution are removed. This removes extreme annotation outliers while preserving valid

hard examples. After filtering, the final dataset contains 19,100 filtered examples, providing dense and precise supervision for PRM training. Overall, SHARP contains a relatively balanced distribution of solution-level correctness: 51% of solutions are entirely correct, while 49% contain at least one error.

4 Experimental Study

4.1 Setup

We fine-tune several publicly available instruction-tuned math LLMs (1.5B–8B), e.g., Qwen2.5-Math-1.5B-Instruct and Qwen2.5-Math-7B-Instruct (Yang et al., 2024), as step-level correctness predictors. We train process reward models following the Qwen2.5-Math-PRM-7B (Zhang et al., 2025) setup, using Qwen2.5-Math-Instruct models as initialization backbones and converting them into process reward models with a dedicated two-way classification head. Specifically, we start from publicly available instruction-tuned math LLMs, add a special step marker token (<extra_0>), and format each training example as a math question followed by a step-by-step solution, where every reasoning step is terminated by this marker. The model is then optimized to predict a binary correctness label at the positions of these step markers only, using cross-entropy loss while masking out all other tokens. After evaluating several PRM backbones (e.g., Llama 3.1 (Grattafiori et al., 2024), Mathstral-7B (Mistral AI, 2024)), we adopted Qwen2.5-Math as our backbone because as shown in Table 3 it consistently outperformed the others across all scenarios.

Following the curriculum learning principle (Bengio et al., 2009), we fine-tune SHARP models in two stages. The PRM is first trained on the GSM8K-derived subset, which contains simpler grade-school mathematical reasoning problems, and is then further fine-tuned on the MATH-derived subset with more challenging and diverse reasoning trajectories. This progression allows the model to first acquire basic step-level correctness patterns before learning to detect more complex mathematical errors.

We employ preprocessed versions of both datasets: for PRM800K, we use a standard variant, with the technical details of the conversion procedure provided in Appendix B.4, while SHARP provides span-level hallucination annotations, which are not directly compatible with step-level super-

Model	GSM8K			MATH			OlympiadBench			Omni-MATH			Avg. F1
	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	
1.5B													
Skywork-o1-PRM-1.5B (He et al., 2024b)	50.2	71.5	59.0	37.9	65.2	48.0	15.4	26.0	19.3	13.6	32.8	19.2	36.4
Qwen2.5-Math-1.5B-SHARP-Step (ours)	36.7	98.9	53.6	52.0	84.7	64.5	47.1	76.4	58.2	47.6	72.6	57.5	58.4
Qwen2.5-Math-1.5B-SHARP-Span (ours)	<u>45.4</u>	99.5	62.4	56.7	<u>82.8</u>	67.3	50.5	<u>69.3</u>	58.5	50.2	<u>71.4</u>	58.9	61.8
7-8B													
RLHFlow-PRM-Deepseek-8B (Xiong et al., 2024)	24.2	98.4	38.8	21.4	80.0	33.8	10.1	51.0	16.9	10.9	51.9	16.9	26.6
RLHFlow-PRM-Mistral-8B (Xiong et al., 2024)	33.8	99.0	50.4	21.7	72.2	33.4	8.2	43.1	13.8	9.6	45.2	15.8	28.4
Qwen2.5-Math-7B-Math-Shepherd (Zhang et al., 2025)	46.4	95.9	62.5	18.9	96.6	31.6	7.4	93.8	13.7	4.0	95.0	7.7	28.9
EurusPRM-Stage1 (Cui et al., 2025)	46.9	42.0	44.3	33.3	38.2	35.6	23.9	19.8	21.7	21.9	24.5	23.1	31.2
EurusPRM-Stage2 (Cui et al., 2025)	51.2	44.0	47.3	36.4	35.0	35.7	25.7	18.0	21.2	23.1	19.1	20.9	31.3
Math-Shepherd-PRM-7B (Wang et al., 2024)	32.4	91.7	47.9	18.0	82.0	29.5	15.0	71.1	24.8	14.2	73.0	23.8	31.5
ReasonEval-7B (Xia et al., 2025)	26.1	95.3	41.0	35.7	77.6	48.9	27.5	55.2	36.7	27.0	60.6	37.4	41.0
Skywork-o1-PRM-7B (He et al., 2024b)	61.8	82.9	70.8	43.8	62.2	53.6	17.9	31.9	22.9	14.0	41.9	21.0	42.1
Qwen2.5-Math-7B-PRM800K (Zhang et al., 2025)	53.1	95.3	68.2	48.0	90.1	62.6	35.7	<u>87.3</u>	50.7	29.8	<u>86.1</u>	44.3	56.5
PURE-PRM-7B (Cheng et al., 2025)	-	-	69.0	-	-	66.5	-	-	48.4	-	-	45.9	57.5
R-PRM-7B-SFT (She et al., 2025)	<u>66.2</u>	92.7	77.2	60.3	88.2	71.6	48.6	77.3	59.6	40.1	75.5	52.3	65.2
RetrievalPRM-7B (Zhu et al., 2025)	64.7	88.1	74.6	67.2	75.6	71.1	56.0	65.2	60.2	52.8	62.7	57.3	65.8
Qwen2.5-Math-7B-SHARP-Step (ours)	54.1	95.9	69.2	61.6	87.4	72.3	53.4	76.4	62.9	48.5	76.4	59.3	65.9
PathFinder-PRM-7B (Pala et al., 2025)	-	-	77.9	-	-	75.3	-	-	65.0	-	-	59.7	69.5
Qwen2.5-Math-7B-SHARP-Span (ours)	63.8	96.9	76.9	68.5	82.8	75.0	<u>57.0</u>	71.7	63.5	58.4	71.0	<u>64.1</u>	69.9
R-PRM-7B-DPO (She et al., 2025)	72.0	91.7	80.7	71.2	83.5	76.9	60.2	67.8	63.8	<u>55.5</u>	65.6	60.1	70.4
Qwen2.5-Math-PRM-7B (Zhang et al., 2025)	72.0	96.4	82.4	68.0	<u>90.4</u>	77.6	55.7	85.5	67.5	55.2	83.0	66.3	73.5

Table 2: ProcessBench step-level error-detection metrics (acc_{err}, acc_{corr}, and F1). Within each parameter group, **bold** marks the best and underline the second-best result per column. Shaded rows denote our models.

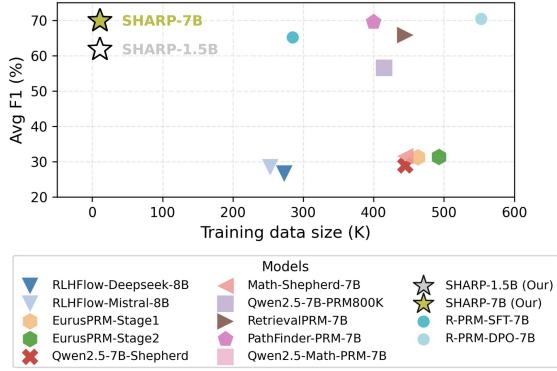


Figure 2: Average F1 Score vs. Training Data Size (K). Our SHARP models achieve state-of-the-art efficiency, reaching competitive F1 scores with only 19.1K instances. Qwen2.5-Math-PRM-7B (1,500K instances) is excluded from the visualization to maintain a legible x-axis scale. Additionally, models with undisclosed training data volumes are omitted.

vision required for training a PRM. We therefore convert the span-level labels into step-level labels as follows.

We mark a solution step as *incorrect* if it contains at least one span annotated as hallucinated. Otherwise, the step is marked as *correct*. Since the SHARP annotations extend to the end of the entire chain of reasoning, we limit them to the first erroneous step and cut the chain of reasoning at the first step with an error (therefore, we exclude the rest from learning). To avoid evaluation leakage, we remove any SHARP instances that overlap with ProcessBench. In this benchmark, models must identify the earliest erroneous step (or confirm that all steps are correct).

4.2 Results

We evaluate all models on ProcessBench (Zheng et al., 2025), which measures step-level error detection quality using classification metrics (acc_{err}, acc_{corr}, and F1), see Table 2.

The results indicate that models finetuned on SHARP demonstrate competitive generalization capabilities across all datasets. Qwen2.5-Math-7B-SHARP-Span achieves an average F1 score of 69.9, representing a 13.4-point absolute improvement over the Qwen2.5-Math-7B-PRM800K baseline (56.5 average F1).

Within the 1.5B parameter category, Qwen2.5-Math-1.5B-SHARP-Span establishes a new state-of-the-art with an average F1 of 61.8. Notably, it outperforms the 7B-parameter PRM800K baseline (56.5 F1) as well as larger 7–8B baselines, including Skywork-o1-PRM-7B (42.1) and Math-Shepherd-PRM-7B (31.5) and others.

4.3 Data Efficiency and Scaling

Figure 2 visualizes the relationship between training data volume and error-detection performance. While a group of contemporary strong PRMs reaches average F1 scores in the 65.0–70.0 range, SHARP-7B achieves a comparable score of 69.9 using significantly fewer training samples. Specifically, our model utilizes only 19K span-annotated instances, which is 10 to 30 times less than the data used by other models in this performance cluster, such as R-PRM-7B (289K (SFT) / 269K (DPO)) / 558K) and RetrievalPRM-7B (445K). This suggests that the dense supervision provided by

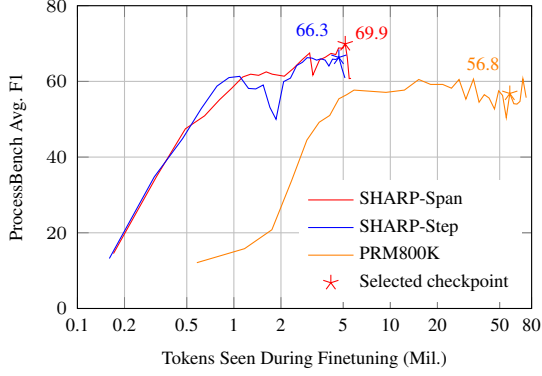


Figure 3: Qwen2.5-Math-7B Model performance (F1) vs Number of Training Tokens. The x -axis represents the number of processed training tokens as a proxy for compute, and the y -axis shows the average F1-score on ProcessBench.

SHARP-Span offers a higher performance-per-sample ratio compared to traditional step-level datasets.

In Figure 3, we compare three training runs that use the same Qwen2.5-Math-7B-Instruct backbone and the same optimization setup, differing only in the training data. SHARP-Span yields the fastest performance gains: its average F1 reaches 47.5 after only 0.49M tokens, exceeds 60 F1 by 1.14M tokens, and reaches the selected checkpoint performance of 69.9 F1 at 5.15M tokens. SHARP-Step follows a similar trajectory but remains consistently below SHARP-Span at the strongest checkpoints, reaching 66.3 F1 at 4.69M tokens. In contrast, PRM800K improves more gradually: it starts at 12.1 F1 after 0.58M tokens, reaches 56.5 F1 at 5.27M tokens, and its selected checkpoint achieves 56.8 F1 only after 58.04M tokens.

Overall, the curve indicates that SHARP-Span is the most computationally efficient choice among the compared training sets. It reaches substantially higher ProcessBench performance within the first few million training tokens, whereas PRM800K requires an order of magnitude more token budget to reach a substantially lower selected-checkpoint score. This makes SHARP-Span especially effective under tight computational budgets and supports the claim that span-derived supervision provides a more efficient training signal for PRMs.

We also compare against size-matched variants of PRM800K. PRM800K-Sampled is limited to the size of the MATH-derived SHARP subset and retains a single reasoning trajectory per question according to Appendix B.4. Restricting

PRM800K substantially improves over the original baseline, increasing Avg. F1 from 56.5 to 63.7, and the PRM800K-Sampled $\rightarrow$ GSM8K curriculum reaches 66.7 Avg. F1. The MATH-only SHARP subset nevertheless remains stronger at 65.4 Avg. F1, and training on it first, followed by GSM8K, reaches 68.9 Avg. F1, outperforming PRM800K-Sampled $\rightarrow$ GSM8K. Note that this is a hard-to-easy ordering, the reverse of the GSM8K $\rightarrow$ MATH curriculum learning principle used for our main models, which attains 69.9 Avg. F1 (Table 2). Although these experiments are limited in scope, they indicate that careful data selection from existing large-scale step-level datasets is a promising direction for improving PRM training efficiency.

4.4 Ablation and Robustness Analysis

Granularity-level ablation. To isolate the effect of annotation granularity, we compare PRMs trained on two controlled variants of the same data source: SHARP-Span and its step-level counterpart, SHARP-Step. In this comparison, the underlying problems, model solutions, backbone family, and training pipeline are fixed, while only the supervision format is varied. As shown in Table 3, the gains of SHARP-Span persist under size-matched training across both Qwen2.5-Math backbones. In both the 1.5B and 7B settings, span-level supervision consistently outperforms step-level supervision, indicating that the finer-grained annotation signal provides a more effective training target than coarse step-level labels. An example of the resulting step-level markup is shown in Figure 8.

Appendix D. gives one case per ProcessBench subset in which SHARP-Step assigns the erroneous step a probability of being correct above 0.93 while SHARP-Span collapses below 0.06.

Backbone robustness. We further examine whether the advantage of SHARP-Span is specific to the Qwen2.5-Math family or holds across different math-oriented LLM backbones. To this end, we repeat the SHARP-Span versus SHARP-Step comparison using Mathstral-7B (Mistral AI, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-Math-7B (Shao et al., 2024), and OpenMath2-Llama3.1-8B (Toshniwal et al., 2025). The same pattern holds across all evaluated backbones: PRMs trained on SHARP-Span achieve better metrics than their SHARP-Step counterparts. This suggests that the benefit of span-derived su-

Model	GSM8K			MATH			OlympiadBench			Omni-MATH			Avg. F1
	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	acc _{err}	acc _{corr}	F1	
Finetuned on SHARP-Span													
Mathstral-7B (Mistral AI, 2024)	41.1	67.4	51.0	35.4	24.9	29.2	27.4	5.3	8.9	33.3	19.1	24.3	28.4
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	7.2	99.7	13.5	24.6	81.3	37.7	32.8	63.1	43.2	31.4	64.3	42.2	34.2
DeepSeek-Math-7B (Shao et al., 2024)	27.1	96.9	42.3	44.4	82.3	57.7	43.7	65.8	52.5	44.7	63.5	52.4	51.2
OpenMath2-Llama3.1-8B (Toshniwal et al., 2025)	30.0	98.4	45.9	41.2	90.4	56.6	43.1	77.0	55.3	43.7	73.0	54.7	53.1
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	45.4	99.5	62.4	56.7	82.8	67.3	50.5	69.3	58.5	50.2	71.4	58.9	61.8
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	63.8	96.9	76.9	68.5	82.8	75.0	57.0	71.7	63.5	58.4	71.0	64.1	69.9
Finetuned on SHARP-Step													
Mathstral-7B (Mistral AI, 2024)	4.3	99.5	8.3	22.9	71.7	34.7	26.9	44.8	33.6	27.0	51.9	35.5	28.1
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	26.6	87.6	40.8	40.9	37.4	39.1	32.7	11.8	17.3	38.2	24.1	29.5	31.7
DeepSeek-Math-7B (Shao et al., 2024)	41.5	94.3	57.7	53.0	61.8	57.1	43.6	29.5	35.2	46.6	39.8	43.0	48.2
OpenMath2-Llama3.1-8B (Toshniwal et al., 2025)	46.9	92.7	62.3	51.9	72.9	60.6	46.7	43.4	45.0	48.7	40.7	44.3	53.0
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	36.7	98.9	53.6	52.0	84.7	64.5	47.1	76.4	58.2	47.6	72.6	57.5	58.4
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	54.1	95.9	69.2	61.6	87.4	72.3	53.4	76.4	62.9	48.5	76.4	59.3	65.9
Finetuned on SHARP-Span (Qwen-gen)													
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	51.7	86.5	64.7	52.2	66.8	58.6	40.5	45.7	43.0	42.0	47.3	44.5	52.7
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	56.0	92.8	69.9	55.1	82.3	66.0	52.5	74.0	61.4	57.4	66.4	61.6	64.7
Finetuned on SHARP-Span (Mistral-gen)													
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	50.7	96.4	66.5	56.7	80.3	66.5	52.7	64.6	58.0	50.7	61.8	55.7	61.7
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	68.1	91.7	78.2	64.5	77.6	70.4	56.1	61.4	58.6	55.6	61.0	58.2	66.4
Finetuned on Alternative span-to-step conversion													
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)	54.6	91.2	68.3	46.5	86.7	60.5	30.4	88.8	45.3	27.9	89.2	42.5	54.2
Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	65.2	93.3	76.8	52.2	89.2	65.8	34.3	90.0	49.7	31.6	92.1	47.1	59.9

Table 3: Performance of PRMs on ProcessBench across different training sets.

pervision is not tied to a particular backbone or preprocessing pipeline, but reflects a more general advantage of finer-grained supervision for process reward modeling.

Generalization across generators. A potential concern is that SHARP-Span is constructed from traces generated by a single model, which may bias both the annotations and the resulting PRMs toward that model’s reasoning style. To examine this, we vary the trace generator while keeping the annotation pipeline fixed. In addition to the original Mathstral-7B-v0.1 traces used in SHARP-Span, we construct parallel variants using Mistral-7B-Instruct (Jiang et al., 2023) and Qwen2.5-Math-7B-Instruct (Yang et al., 2024), with gpt-oss-120B providing span-level error annotations in all cases. PRMs trained on Mistral-7B-Instruct-generated traces achieve average F1 comparable to the original SHARP-Span setting, while those trained on Qwen2.5-Math-7B-Instruct-generated traces remain competitive, trailing by only 5.2 points for the 7B backbone and 9.1 points for the 1.5B backbone. Overall, these results suggest that the benefits of span-level supervision are not tied to a single generator model, al-

though generator choice still has a measurable effect on downstream PRM quality. In Table 3, the blocks SHARP-Span (Qwen-gen) and SHARP-Span (Mistral-gen) denote SHARP-Span variants whose reasoning traces were produced by Qwen2.5-Math-7B-Instruct and Mistral-7B-Instruct, respectively, while the unlabeled SHARP-Span block uses the original Mathstral-7B-v0.1 traces.

First-error truncation ablation. Finally, we examine whether training on the full reasoning trace after the first erroneous step improves PRM performance. Our main preprocessing follows the ProcessBench evaluation protocol: after converting span-level annotations to step-level labels, we truncate each trajectory at the first erroneous step and exclude all subsequent steps from training. We compare this strategy against a variant that keeps the full chain of reasoning, including steps after the first detected error.

Keeping the post-error suffix does not improve performance and consistently leads to worse downstream metrics. This result suggests that steps following the first error provide an ambiguous supervision signal. Once the reasoning state becomes invalid, later steps may be algebraically or logically

consistent with the incorrect premise, may merely propagate the initial error, or may even attempt to self-correct. As a result, their labels become less directly tied to the objective of detecting the earliest erroneous step. Truncating at the first error removes this ambiguity and better aligns the training objective with ProcessBench, in which the model is forced to identify the first incorrect step.

As shown in Table 3, the alternative span-to-step conversion that keeps post-error reasoning substantially underperforms the main first-error truncation strategy. For Qwen2.5-Math-1.5B, Avg. F1 drops from 61.8 to 54.2, while for Qwen2.5-Math-7B it drops from 69.9 to 59.9.

4.5 Analysis of Dataset Construction Choices

Beyond model-level ablations, we analyze several design choices in the SHARP construction pipeline. These experiments examine how the choice of source problem datasets, class balance, annotator decoding parameters and filtering strategy affect the quality of the resulting PRM supervision.

Problem Sources and Curriculum Order. We analyze the choice of source problem datasets by evaluating GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and Omni-MATH (Gao et al., 2024), which represent increasing levels of mathematical difficulty, together with their pairwise and three-way combinations. For each composition, we consider both easy-to-hard and hard-to-easy curriculum orders. The best overall performance is achieved by the GSM8K + MATH combination, which outperforms other dataset mixtures regardless of training order. By contrast, configurations that include Omni-MATH consistently hurt performance — its reference solutions contain unusual patterns, including Python-based proofs and inconsistent annotations, that degrade downstream PRM quality.

Class-balance ablation. We further study how the balance between correct and incorrect step-level labels affects PRM training. Since the raw generated traces produce a naturally imbalanced supervision signal, we construct several controlled training subsets with different incorrect-to-correct step ratios, ranging from 1:9 to 1:1. To obtain these subsets, we first generate multiple candidate reasoning traces per problem and then select trajectories so that the resulting step-level label distribution matches the target ratio. During this process, we apply question-level de-duplication, ensuring

that each final training subset contains only unique problems and that the observed differences are not caused by repeated questions.

As shown in Table 7, the optimal class balance depends on the training stage and dataset difficulty. For the first-stage GSM8K training, the best validation performance is obtained with an incorrect-to-correct ratio of 3:7. In contrast, during the second curriculum stage on the MATH-derived subset, where the model is initialized from the best GSM8K checkpoint, the best performance is achieved with a more skewed ratio of 1:9. This suggests that early PRM training benefits from a stronger negative supervision signal, while subsequent fine-tuning on harder mathematical problems requires a more conservative amount of erroneous steps to avoid over-representing failures and distorting the training distribution.

More generally, these results indicate that class balancing should not be treated as a fixed global hyperparameter. Instead, the optimal ratio depends on both the difficulty of the source dataset and the position of that dataset in the curriculum. Moderate rebalancing helps the model learn to detect reasoning errors, but excessive exposure to incorrect steps can reduce generalization to ProcessBench-style evaluation.

Annotator decoding ablation. We also evaluate the effect of decoding parameters used by the LLM-as-a-judge annotator. In particular, we compare a deterministic configuration, temperature $T = 0$ with top-p = 1.0, against a more stochastic configuration, temperature $T = 0.7$ with top-p = 0.8. Across all runs, the deterministic setting achieves better average downstream PRM performance, improving Avg. F1 by 0.3 points compared to the stochastic configuration.

We hypothesize that this effect comes from the nature of the annotation task. Span-level hallucination annotation requires stable token-level decisions and strict preservation of the original reasoning trace. Sampling-based decoding can introduce variability in boundary selection, missed spans, or malformed outputs, which increases label noise. Therefore, we use deterministic decoding for the final SHARP annotation pipeline.

Filtering ablation. We compare three filtering strategies for the span-level annotations: no filtering, 97.5th-percentile filtering, and 95th-percentile filtering. Both percentile-based variants remove annotation outliers according to empirical span statis-

tics, such as hallucination span length and the total number of marked fragments per solution.

The unfiltered dataset contains noisy generations and malformed annotations that degrade downstream PRM training. The 97.5th-percentile cutoff improves over the unfiltered setting, but it remains too permissive: many noisy annotations and malformed samples are retained, which continues to introduce label noise into PRM training. In contrast, the 95th-percentile strategy provides a better trade-off by removing a larger portion of harmful annotation outliers while preserving valid hard examples. Empirically, 95th-percentile filtering improves downstream PRM performance by 2.1 Avg. F1 points compared to training on the unfiltered dataset and by 1.2 Avg. F1 points compared to the 97.5th-percentile variant. Based on this ablation, we adopt 95th-percentile filtering in the final dataset construction pipeline.

4.6 Annotation Quality and Human-LLM Agreement

To rigorously evaluate the quality of our generated hallucination dataset and assess the reliability of the Large Language Model (LLM) as an automatic annotator, we conducted a human evaluation. From the main dataset, a subset of 500 generated mathematical solutions was sampled. The validation process was divided into two phases: assessing Inter-Annotator Agreement (IAA) among humans, and evaluating the alignment between human annotators and the LLM.

Although our dataset provides fine-grained, span-level annotations of mathematical hallucinations, our ultimate objective is to utilize this data to train a PRM, which evaluates reasoning on a step-by-step basis. Consequently, we report metrics at two levels of granularity:

- **Token-level:** Evaluates the exact overlap of the highlighted hallucination spans.
- **Step-level:** Evaluates whether the annotators flagged the same logical reasoning step (e.g., an equation line or a paragraph) as containing a hallucination, regardless of the exact token boundaries.

Inter-Annotator Agreement (IAA) To establish a human ground truth, 100 samples were independently annotated by four expert human annotators. Table 4 presents the agreement metrics compared against the Majority Vote.

Annotator	Token-level F1	Step-level F1	Step-level Kappa (κ)
Annotator A	0.773	0.837	0.795
Annotator B	0.759	0.840	0.797
Annotator C	0.658	0.845	0.804
Annotator D	0.758	0.831	0.786
Human Avg.	0.737	0.838	0.796

Table 4: Inter-Annotator Agreement on the shared subset (100 samples) compared against the Majority Vote. Step-level metrics demonstrate substantial consensus on identifying erroneous reasoning steps.

While Token-level F1 shows expected natural variation in how humans select text boundaries (e.g., including or excluding mathematical formatting symbols like \$), the Step-level metrics demonstrate high consistency. The average human Step-level Cohen’s Kappa of 0.796 indicates near-perfect agreement on the localization of mathematical errors.

Human vs. LLM Alignment The remaining 400 samples were divided into four disjoint sets, each annotated by one human expert and the LLM. Furthermore, we evaluated the LLM against the human Majority Vote on the shared 100-sample subset. Table 5 summarizes the alignment between the LLM’s automatic <HAL> tagging and human judgments. We also report **First Error Accuracy**, which measures how often the LLM and humans agree on the exact step of the initial hallucination.

Data Split	Token F1	Step F1	Step Kappa (κ)	First Err. Acc.
Subset 1	0.551	0.812	0.718	89.0%
Subset 2	0.430	0.847	0.589	77.0%
Subset 3	0.431	0.840	0.613	76.0%
Subset 4	0.664	0.890	0.649	85.0%
Subset 5	0.571	0.847	0.597	—

Table 5: Alignment metrics between the LLM hallucination annotator and human annotators. Subsets 1-4 represent pairwise Human-LLM comparisons, while Subset 5 compares the LLM against the human Majority Vote. First Error Accuracy indicates agreement on the exact initial failing step.

The results validate the use of the LLM for large-scale hallucination detection. Although the LLM exhibits a specific stylistic bias in token boundary selection (yielding a moderate Token-level F1 of ~ 0.52), it is highly proficient at identifying the correct underlying mathematical errors. The LLM maintains balanced precision and recall at the step level, achieving an F1 score ranging from 0.812 to 0.890. More importantly, the LLM identifies the exact *first* erroneous step with an average accuracy

of 81.8% across subsets 1-4. This level of functional accuracy indicates that the auto-generated annotations are robust enough to serve as reliable step-level supervision for training PRMs.

Cross-Annotator Robustness The analysis above validates the LLM annotator against human experts. A separate question is whether the resulting labels depend on the choice of annotator model. We re-annotated all SHARP traces with a second annotator from a different model family, Qwen3-235B-A22B-Instruct-2507 (Qwen Team, 2025a,b), holding the traces, the prompt, the decoding configuration ($T = 0$), and the filtering pipeline fixed; the annotator model is the only varied factor.

Granularity	Agreement	κ
Solution-level (has ≥ 1 error)	95.0%	0.899
Step-level	90.7%	0.686
Span-level (character F1)	0.689	—
First error (exact)	66.1%	—
First error (± 1 step)	76.9%	—

Table 6: Agreement between the two automatic annotators, gpt-oss-120B and Qwen3-235B-A22B-Instruct-2507, over the full SHARP dataset.

Table 6 reports agreement between the two annotators. At the solution level the two annotators agree at 95.0% ($\kappa = 0.899$), and at the step level at 90.7% ($\kappa = 0.686$). The human study in Table 4 reports a step-level κ of 0.796, but that figure is computed against the majority vote of four annotators rather than between annotator pairs, and on 100 samples rather than the full dataset; the two are therefore indicative rather than directly comparable. We likewise report character-level span F1 (0.689) alongside, but do not equate it to, the token-level figures in Tables 4–5, since the units differ. Exact first-error agreement is 66.1%. As in the human study, it degrades with problem difficulty (74.6% on MATH Level 1 versus 51.5% on Level 5), which we attribute to genuine ambiguity in where a chain of reasoning first breaks rather than to annotator unreliability.

The two annotators induce comparable supervision statistics: the median span length is identical (3.0 versus 3.0), while the gpt-oss annotator marks somewhat more spans per solution (2.93 versus 2.31) and produces fewer malformed outputs (0.3% versus 1.6%). An independently chosen annotator from a different family therefore reproduces the labels at the solution and step level, with only

moderate differences in span density.

We stress the scope of this result: it establishes that the labels do not depend on idiosyncrasies of a single annotator model, not that they are correct. Two LLM annotators may share systematic biases that human annotation would expose, and cross-model agreement is therefore complementary to, not a substitute for, the human validation above.

5 Conclusion

We presented SHARP, the first open-source dataset providing fully automatic span-level hallucination annotations for model-generated mathematical chain-of-thought reasoning. Our experiments show that step-level labels derived from span annotations train highly data-efficient PRMs on ProcessBench: SHARP establishes a new state of the art among 1.5B PRMs and achieves top-three performance among 7–8B PRMs, coming within 0.5 Avg. F1 of the second-best model while using $29\times$ fewer training instances. Our ablation studies further highlight the value of span-level granularity over simple step-level labels. In addition, our limited experiments with PRM800K-Sampled indicate that filtering existing large-scale step-level datasets can substantially improve PRM quality, highlighting data cleaning and selection as an additional direction for improving training efficiency in future work. While our work focuses on mathematical reasoning, SHARP provides a practical framework for developing data-efficient, fine-grained verifiers.

Limitations

Our work has several limitations. The final SHARP dataset is constructed from GSM8K and MATH, and therefore remains limited to mathematical reasoning problems. All samples were generated by a single small model, which introduces bias toward errors characteristic of that specific model. We plan to extend the dataset to cover additional tasks and to incorporate multiple models for hallucination generation.

Although we conduct a human evaluation with multiple expert annotators, the validation set remains relatively small compared to the full dataset. Future work should expand this evaluation to a larger and more diverse sample of problems, annotators, and mathematical domains.

Finally, although SHARP provides span-level annotations, our downstream evaluation is step-level. ProcessBench defines errors at the step level,

and no public span-level benchmark for mathematical reasoning currently exists. We therefore validate the span annotations themselves against human experts (Section 4.6), but we do not directly measure how accurately the trained PRMs localize erroneous spans; our results should be read as evidence that span-level supervision yields better step-level verifiers. Constructing a human-labelled span-level test set is left to future work.

Ethical Considerations

A potential risk is that users may over-rely on the model’s error detection despite possible false positives or false negatives. We mitigate this by clearly documenting its limitations and intended research use. As with any open-source release, unintended downstream use is possible, which we address through usage guidelines. Overall, the risks are limited and outweighed by the benefits for research on reasoning robustness and error analysis.

The dataset does not contain any personal or sensitive information. It consists exclusively of fully model-generated mathematical problem solutions with annotations marking incorrect reasoning steps. No data is derived from human users, and no personally identifiable information is collected, included, or inferred.

Acknowledgments

This work was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, subsidy identifier 000000C313925P4B0002).

References

- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. [Curriculum learning](#). In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada. ACM Press.
- Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue Wang. 2025. [Stop summation: Min-form credit assignment is all process reward model needs for reasoning](#). *CoRR*, abs/2504.15275.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, and 6 others. 2025. [Process reinforcement through implicit rewards](#). *CoRR*, abs/2502.01456.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. [Omni-MATH: A universal olympiad level mathematic benchmark for large language models](#). *CoRR*, abs/2410.07985.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and 1 others. 2024. [The Llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024a. [OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems](#). 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 3828–3850. Association for Computational Linguistics.
- Jujie He, Tianwen Wei, Rui Yan, Jiakai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024b. [Skywork-o1 Open Series](#). Zenodo, version 1.0.0.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7B](#). *CoRR*, abs/2310.06825.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

- Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. 2020. [On faithfulness and factuality in abstractive summarization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1906–1919. Association for Computational Linguistics.
- Mistral AI. 2024. [Mathstral-7B-v0.1](https://huggingface.co/mistralai/Mathstral-7B-v0.1). <https://huggingface.co/mistralai/Mathstral-7B-v0.1>. Hugging Face model card.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). *CoRR*, abs/2508.10925.
- Tej Deep Pala, Panshul Sharma, Amir Zadeh, Chuan Li, and Soujanya Poria. 2025. [Error typing for smarter rewards: Improving process reward models with error-aware hierarchical supervision](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14940–14954, Suzhou, China. Association for Computational Linguistics.
- Qwen Team. 2025a. [Qwen3-235B-A22B-Instruct-2507](#). Hugging Face model card.
- Qwen Team. 2025b. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *CoRR*, abs/2402.03300.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. [R-PRM: Reasoning-driven process reward modeling](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451, Suzhou, China. Association for Computational Linguistics.
- Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanian, Alexan Ayrapetyan, and Igor Gitman. 2025. [OpenMathInstruct-2: Accelerating AI for math with massive open-source instruction data](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *CoRR*, abs/2211.14275.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. [Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 9426–9439. Association for Computational Linguistics.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. [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.
- Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. [An implementation of generative PRM](#). <https://github.com/RLHFlow/RLHF-Reward-Modeling>.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. [Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement](#). *CoRR*, abs/2409.12122.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. [The lessons of developing process reward models in mathematical reasoning](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10495–10516. Association for Computational Linguistics.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. [ProcessBench: Identifying process errors in mathematical reasoning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 1009–1024. Association for Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena](#). 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*.
- Jiachen Zhu, Congmin Zheng, Jianghao Lin, Kounianhua Du, Ying Wen, Yong Yu, Jun Wang, and Weinan

Zhang. 2025. [Retrieval-augmented process reward model for generalizable mathematical reasoning](#). In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 8453–8468. Association for Computational Linguistics.

Appendix A. Hyperparameters

We train with global batch size 128, gradient accumulation 1, learning rate $2 \cdot 10^{-5}$, cosine learning-rate schedule, warmup ratio 0.03, bf16 mixed precision, and gradient checkpointing. We use early stopping with patience 3 and a threshold of 0.01, selecting the best checkpoint based on an F1 metric.

Appendix B. Additional experimental details

B.1 Computational Resources

All dataset generation and model training were conducted on a single machine equipped with four NVIDIA A100 GPUs with 80 GB of memory each. For the final SHARP models, training a 1.5B PRM required approximately 15 minutes for the GSM8K stage and 25 minutes for the MATH stage, which we conservatively round to about 1 wall-clock hour on $4 \times$ A100 GPUs, or approximately 4 GPU-hours. Training a 7B PRM required approximately 1 hour for the GSM8K stage and 2 hours for the MATH stage, for a total of about 3 wall-clock hours on $4 \times$ A100 GPUs, or approximately 12 GPU-hours.

Across the training runs reported in the main results and ablation tables, including the PRM800K and PRM800K-Sampled comparison runs, the total computational budget was approximately 210–220 GPU-hours.

B.2 Backbones

In addition to Qwen2.5-Math-1.5B-Instruct and Qwen2.5-Math-7B-Instruct (Yang et al., 2024), we also fine-tune Mathstral-7B (Mistral AI, 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-Math-7B (Shao et al., 2024), and OpenMath2-Llama3.1-8B (Toshniwal et al., 2025) under the same PRM training recipe. The full ProcessBench metrics for all ablation conditions are reported in Table 3.

B.3 Data format

The training dataset contains (at minimum) three fields: question (problem statement), steps (a list of reasoning steps), and step_labels (a list

of step-level labels aligned with steps, e.g., 0/1 values indicating whether each step is correct).

B.4 PRM800K Preprocessing

PRM800K contains 101,599 labeled solution trajectories collected in two phases. Phase 1 labels multiple alternative continuations at each step, which helps seed the dataset but introduces substantial redundancy and annotation artifacts, accounting for approximately 5% of PRM800K (roughly 40,000 step-level labels). Phase 2 scales up and streamlines labeling by pre-generating complete solutions and stopping annotation at the first negative step; alternative completions are mainly collected after the first identified mistake. In our work, we use only Phase 2 due to the additional complexities and implicit conventions present in Phase 1.

One-trajectory-per-sample conversion. In the original PRM800K format, a single record may contain multiple alternative completions for an erroneous step. We convert the dataset into a denormalized format where one sample corresponds to exactly one complete reasoning trajectory. After this conversion, we obtain 404,466 training samples and 10,528 test samples.

Filtering and de-duplication. We remove samples with incomplete supervision (e.g., cases where annotation stops at an error step but some alternative completions at that step are rated as correct, leaving the remaining steps unlabeled), as they introduce ambiguous targets for step-level training. We also remove samples with `finish_reason` set to `give_up` or `bad_problem`, because these instances contain missing ratings and lack reliable annotations. After these filters, we obtain 245,040 training samples and 6,318 test samples. After additional de-duplication and cleaning, the dataset contains 208,613 training samples and 6,318 test samples.

Decontamination against ProcessBench. To prevent evaluation leakage, we remove any samples that overlap with those in ProcessBench. The final processed dataset contains 196,968 training samples and 4,890 test samples.

Appendix C. Ablation study details

SHARP-Step. We evaluate agreement between our span-level annotations and step-level labels produced by gpt-oss-120B. We reuse the same Mathstral-7B-v0.1 responses and append a [STEP_END] marker to each step (steps are sepa-

rated by double newlines, “\n\n”). The prompt (Appendix Fig. 6) instructs the model to identify steps solely via these markers and assign [HAL] or [NO HAL]. We discard malformed generations and responses longer than 2,048 tokens, following the same preprocessing procedure as for SHARP-Span.

Results. Table 3 reports the full ProcessBench metrics for all ablation conditions.

Appendix D. Case study: span- versus step-level supervision

For each ProcessBench subset we select one solution in which the ground-truth first erroneous step is otherwise entirely valid—the formula, the derivation, and the arithmetic around the defect are all correct—so that a holistic step-level verdict is dominated by the surrounding valid content. Probabilities are $p(\text{step is correct})$ as predicted by the Qwen2.5-Math-7B-SHARP-Span and Qwen2.5-Math-7B-SHARP-Step PRMs of Table 2.

GSM8K (error at step 3). *Problem:* It takes James 10 minutes to read 3 pages. He reads 18 pages. How long does he spend reading? *Erroneous step:* “We now multiply the total number of pages (18) by his reading rate (0.3 pages per minute). Total time = $18 \times 0.3 = 5.4$ minutes.” *Defect:* a single wrong operator—the rate must be divided, not multiplied. The rate itself was derived correctly in the preceding step. *SHARP-Span:* 0.010, flags step 3. *SHARP-Step:* 0.996, no error predicted.

MATH (error at step 2). *Problem:* A right hexagonal prism has height 3 feet; each edge of the hexagonal base is 6 inches. Find the sum of the areas of the non-hexagonal faces. *Erroneous step:* “Each rectangular face has a height of 3 feet and a width equal to the perimeter of the hexagonal base, which is 3 feet. Area = height $\times$ width = $3 \times 3 = 9$ sq ft.” *Defect:* a single wrong substitution—the face width is the hexagon’s edge (6 inches), not its perimeter. The formula and the arithmetic are internally consistent. *SHARP-Span:* 0.006, flags step 2. *SHARP-Step:* 1.000, no error predicted.

OlympiadBench (error at step 10). *Problem:* $T = 8/5$, $z_1 = 15 + 5i$, $z_2 = 1 + Ki$. Find the smallest positive integer K such that $|z_1 - z_2| \geq 15T$. *Erroneous step:* “Approximate $\sqrt{380}$ as $\sqrt{361} = 19$ (since $\sqrt{380}$ lies between $\sqrt{361} = 19$ and $\sqrt{400} = 20$): $K \geq 5 + 19 = 24$.” *Defect:* a single unjustified approximation— $\sqrt{380}$

is replaced by a smaller value, invalidating the bound. The preceding modulus derivation is correct. *SHARP-Span:* 0.055, flags step 10. *SHARP-Step:* 0.934, no error predicted.

Omni-MATH (error at step 13). *Problem:* Find all integer triples (a, b, c) satisfying $a + b = c$ and $a^2 + b^3 = c^2$. *Erroneous step:* “Since $8a$ is even, k must also be odd ... Let $k = 2m + 1$, so $8a = 4m(m + 1)$, hence $a = m(m + 1)$.” *Defect:* a single invalid premise—the parity inference does not follow. The algebra that follows is executed correctly on a false premise. *SHARP-Span:* 0.030, flags step 13. *SHARP-Step:* 0.977, no error predicted.

In all four cases the defect is a minimal local element that step-level supervision masks. We note that these are illustrative cases selected to characterise the failure mode, not a quantitative evaluation of span localization.

Backbone: Qwen2.5-Math-7B-Instruct							
Stage	Training data	Ratio	ProcessBench F1				Avg. F1
			GSM8K	MATH	OlympiadBench	Omni-MATH	
Stage 1	GSM8K	50/50	75.7	63.6	53.8	52.5	61.4
	GSM8K	40/60	74.9	64.4	55.5	51.3	61.5
	GSM8K	30/70	75.7	65.5	54.4	50.7	61.6
	GSM8K	20/80	74.0	63.8	54.5	52.0	61.1
	GSM8K	10/90	61.0	53.8	42.7	35.1	48.2
Stage 2	GSM8K → MATH	50/50	80.4	69.1	56.8	56.8	65.8
	GSM8K → MATH	40/60	75.9	72.3	59.4	59.0	66.6
	GSM8K → MATH	30/70	78.2	72.9	61.6	59.9	68.2
	GSM8K → MATH	20/80	76.4	71.8	61.6	61.7	67.9
	GSM8K → MATH	10/90	76.9	75.0	63.5	64.1	69.9

Table 7: ProcessBench F1 results under two-stage fine-tuning. In Stage 1, Qwen2.5-Math-7B-Instruct is fine-tuned on the GSM8K-derived subset. In Stage 2, all models are initialized from the best Stage 1 checkpoint and further fine-tuned on the MATH-derived subset. The ratio denotes the incorrect-to-correct step-level label ratio used at the corresponding training stage. Bold values indicate the best-performing model within each stage according to Avg. F1.

Appendix E. Prompts used in pipeline

You are a helpful math assistant. Solve problems step by step, then provide the final answer without extra comments.
 Problem:
 {problem}
 Answer:

Figure 4: Prompt used in answers generation

You are given:

- A math problem
- A correct reference solution
- A model’s solution that may contain calculation or reasoning errors

Your task:

1. Compare the model’s solution with the reference solution **step by step**
2. Identify **ONLY** factual mathematical errors:
 - Wrong numerical results
 - Incorrect formulas or transformations
 - Invalid logical or reasoning steps
 - False mathematical statements
3. For each specific erroneous **symbol, number, operator, or word**, wrap it in `<HAL></HAL>` tags.
4. Keep all correct parts of the model’s reasoning exactly as they are.
5. **Minimize each span** - tag only the smallest possible incorrect element (e.g., a single digit, operator, or term), not full sentences or equations.

Tagging policy:

- Wrap only the minimal incorrect token(s) - e.g., `<HAL>12</HAL>` instead of `<HAL>2x = 12</HAL>`
- Use multiple small spans instead of one large one
- Merge spans only if errors are directly related (e.g., both parts of the same incorrect equation)
- Do NOT wrap correct context, explanation, or valid intermediate steps
- Do NOT alter formatting or structure of the model’s answer
- Do NOT provide any commentary or evaluation

```

### What counts as a hallucination in math:
- Incorrect numerical computation
- Wrong mathematical formula or operation
- Invalid or illogical reasoning step
- False statement about a property or result
- Incorrect final answer

### What does NOT count as a hallucination:
- Alternative valid reasoning paths
- Equivalent algebraic forms
- Different (but correct) notation or formatting

### Input:
Question: {problem}

Reference Answer: {solution}

Model's Answer: {model_answer}

### Output requirements:
- Return the full model answer, but with <HAL></HAL> tags around every
  minimal erroneous element.
- Preserve the exact original structure, spacing, and wording of the model's
  answer.
- If **no errors** are detected, output exactly: No hallucinations

### Example:
Input:
Question: Solve  $2x + 5 = 15$ 
Reference:  $2x = 10$ ,  $x = 5$ 
Model's Answer:  $2x + 5 = 15$ , so  $2x = 12$ , therefore  $x = 6$ 

Output:
 $2x + 5 = 15$ , so  $2x =$  <HAL>12</HAL>, therefore  $x =$  <HAL>6</HAL>

```

Figure 5: Prompt used in span-level annotation

```

You are given:
- A math problem
- A correct reference solution
- A model's solution that may contain calculation or reasoning errors

Your task:
1. Compare the model's solution with the reference solution step by step.
2. A step is defined as any text that ends with the explicit marker [
  STEP_END].
   - The marker [STEP_END] explicitly denotes the end of a step.
   - Every occurrence of [STEP_END] corresponds to exactly one step.
3. For every step decide:
   - [HAL] if the step contains at least one mathematical or logical error
   - [NO HAL] if the step is fully correct
4. Preserve the original text and formatting of each step exactly, including
   the [STEP_END] marker.
5. Do not rewrite, comment, summarize, or modify the content.
6. Errors include only factual mathematical errors:
   - Incorrect numbers or calculations
   - Invalid formulas or transformations
   - False mathematical statements
   - Invalid reasoning or logic
7. Do not classify as errors:
   - Equivalent formulations
   - Alternative valid reasoning
   - Differences in notation

```

```

### Output format:
Return the model's answer step by step.
For each step:
- Output the original step text exactly as is (ending with [STEP_END])
- Immediately after the [STEP_END] marker, append exactly one label: [HAL]
  or [NO HAL]

No extra commentary.

### Input:
Math Problem: {problem}
Reference Solution: {solution}
Model's Solution: {model_answer}

### Output example:

Input:
Math Problem: Solve  $2x + 5 = 15$ 
Reference Solution:  $2x = 10, x = 5$ 
Model's Solution:
 $2x + 5 = 15$ . [STEP_END]
 $2x = 12$ . [STEP_END]
Therefore  $x = 6$ . [STEP_END]

Output:
 $2x + 5 = 15$ . [STEP_END] [NO HAL]
 $2x = 12$ . [STEP_END] [HAL]
Therefore  $x = 6$ . [STEP_END] [HAL]

```

Figure 6: Prompt used in step-level annotation

Appendix F. Dataset samples

```

{
  "problem": "Let  $f(x) = \begin{cases} ax+3, & \text{if } x > 2, \\ x-5 & \text{if } -2 \leq x \leq 2, \\ 2x-b & \text{if } x < -2. \end{cases}$  Find  $a+b$  if the piecewise function is continuous (which means that its graph can be drawn without lifting your pencil from the paper).",
  "solution": "For the piecewise function to be continuous, the cases must \"meet\" at  $x=2$  and  $x=-2$ . For example,  $ax+3$  and  $x-5$  must be equal when  $x=2$ . This implies  $a(2)+3=2-5$ , which we solve to get  $2a=-6 \implies a=-3$ . Similarly,  $x-5$  and  $2x-b$  must be equal when  $x=-2$ . Substituting, we get  $-2-5=2(-2)-b$ , which implies  $b=3$ . So  $a+b=-3+3=\boxed{0}$ .",
  "model_answer": "To find the values of  $a$  and  $b$  such that the piecewise function is continuous, we need to ensure that the function values match at the boundaries. At  $x=2$ , the function values should be the same:  $a(2)+3 = 2-5 \implies a = -7$ . At  $x=-2$ , the function values should also be the same:  $2(-2)-b = -2 \implies b = 8$ . So,  $a+b = -7 + 8 = 1$ .",
  "hallucination_annotation": "To find the values of  $a$  and  $b$  such that the piecewise function is continuous, we need to ensure that the function values match at the boundaries. At  $x=2$ , the function values should be the same:  $a(2)+3 = 2-5 \implies a = -7$ . At  $x=-2$ , the function values should also be the same:  $2(-2)-b = -2 \implies b = 8$ . So,  $a+b = -7 + 8 = 1$ ."
}

```

Figure 7: SHARP Span-level sample

```
{
  "problem": "What is the sum of all of the solutions of the equation  $\frac{4x}{20} = \frac{5}{x}$ ?",
  "solution": "Rewrite  $\frac{4}{20}$  as  $\frac{1}{5}$  and multiply both sides by  $5x$  to obtain  $x^2 = 25$ . The solutions of this equation are  $\pm\sqrt{25} = \pm 5$ , and their sum is  $(-5) + 5 = \boxed{0}$ .",
  "model_answer": "First, cross-multiply both sides of the equation to eliminate the fractions:[STEP_END]\n\n $4x * x = 5 * 20$ [STEP_END]\n\nThis simplifies to:[STEP_END]\n\n $4x^2 = 100$ [STEP_END]\n\nNow, divide both sides by 4 to isolate  $x^2$ : [STEP_END]\n\n $x^2 = \frac{100}{4}$ [STEP_END]\n\nNext, take the square root of both sides to find the solutions for  $x$ : [STEP_END]\n\n $x = \pm\sqrt{\frac{100}{4}}$ [STEP_END]\n\nSimplify the square root:[STEP_END]\n\n $x = \pm\sqrt{25}$ [STEP_END]\n\nFinally, since the square root of 25 has two solutions,  $x = \pm 5$ . [STEP_END]\n\nTo find the sum of all solutions, add the two values: [STEP_END]\n\n $x_1 + x_2 = 5 + (-5) = 0$ [STEP_END]\n\nSo, the sum of all solutions of the equation is 0. [STEP_END]",
  "hallucination_annotation": "First, cross-multiply both sides of the equation to eliminate the fractions:[STEP_END] [NO HAL]\n\n $4x * x = 5 * 20$ [STEP_END] [NO HAL]\n\nThis simplifies to:[STEP_END] [NO HAL]\n\n $4x^2 = 100$ [STEP_END] [NO HAL]\n\nNow, divide both sides by 4 to isolate  $x^2$ : [STEP_END] [NO HAL]\n\n $x^2 = \frac{100}{4}$ [STEP_END] [NO HAL]\n\nNext, take the square root of both sides to find the solutions for  $x$ : [STEP_END] [NO HAL]\n\n $x = \pm\sqrt{\frac{100}{4}}$ [STEP_END] [NO HAL]\n\nSimplify the square root:[STEP_END] [NO HAL]\n\n $x = \pm\sqrt{25}$ [STEP_END] [NO HAL]\n\nFinally, since the square root of 25 has two solutions,  $x = \pm 5$ . [STEP_END] [NO HAL]\n\nTo find the sum of all solutions, add the two values: [STEP_END] [NO HAL]\n\n $x_1 + x_2 = 5 + (-5) = 0$ [STEP_END] [NO HAL]\n\nSo, the sum of all solutions of the equation is 0. [STEP_END] [NO HAL]"
}
```

Figure 8: SHARP Step-level sample

Appendix G. Responsible Research Practices

G.1 Licenses for External Artifacts

All external datasets and model checkpoints used in this work are publicly available and used in accordance with their respective licenses. GSM8K, MATH, and PRM800K are released under the MIT License. ProcessBench and Omni-MATH are released under the Apache 2.0 License. Mathstral-7B-v0.1, Mistral-7B-Instruct, Qwen2.5-Math-1.5B-Instruct, and Qwen2.5-Math-7B-Instruct are released under the Apache 2.0 License. The two automatic annotators are likewise released under the Apache 2.0 License: gpt-oss-120B, whose weights are additionally subject to the gpt-oss usage policy, and Qwen3-235B-A22B-Instruct-2507. DeepSeek-Math-7B is released under the DeepSeek Model License, which permits research use. OpenMath2-Llama3.1-8B and Llama-3.1-8B-Instruct are released under the Llama 3.1 Community License. We use all external artifacts for non-commercial research purposes, consistent with their intended use. We release the reasoning traces and hallucination annotations of SHARP under the CC BY 4.0 License, and the SHARP PRM checkpoints under the Apache 2.0 License, following their base models.

G.2 Intended Use of External Artifacts

All external datasets (GSM8K, MATH, PRM800K) are used for research purposes, consistent with their intended use as benchmarks and training resources for mathematical reasoning. All model checkpoints (Mathstral-7B-v0.1, Qwen2.5-Math-1.5B/7B-Instruct, DeepSeek-Math-7B, OpenMath2-Llama3.1-8B, Llama-3.1-8B-Instruct) are used as backbone initializations for fine-tuning and evaluation, consistent with their intended research use. The SHARP dataset released as part of this work is intended for research on hallucination detection and process reward modeling. gpt-oss-120B and Qwen3-235B-A22B-Instruct-2507 are used as automatic annotators for span-level error markup, and Mistral-7B-Instruct and Qwen2.5-Math-7B-Instruct as alternative trace generators, all consistent with their intended research use.

G.3 Human Annotation

The human evaluation study (Section 4.6) was conducted by four expert annotators, all of whom are co-authors of this paper. Annotators were given

the same annotation guidelines used in the automated pipeline (Appendix E.): they were asked to compare model-generated solutions against reference solutions and mark erroneous spans with HAL tags. As the annotators are co-authors, no external recruitment or payment was involved, and no separate consent procedure or ethics board approval was required beyond standard authorship responsibilities.