Efficient PRM Training Data Synthesis via Formal Verification

Ryo Kamoi[1] **Yusen Zhang**[2] **Nan Zhang**[1] **Sarkar Snigdha Sarathi Das**[1]
Ranran Haoran Zhang[1] **Wenpeng Yin**[1] **Rui Zhang**[1]
[1]Penn State University [2]Columbia University
{ryokamoi, rmz5227}@psu.edu

Abstract

Process Reward Models (PRMs) have emerged as a promising approach for improving LLM reasoning capabilities by providing process supervision over reasoning traces. However, existing approaches for constructing PRM training data remain costly and noisy, as they typically rely on human annotation or sampling-based labeling methods that require repeated LLM calls. In this work, we propose FoVer, a framework that synthesizes PRM training data from formal reasoning tasks by annotating step-level error labels using formal verification tools such as Z3 and Isabelle. By leveraging formal verification, FoVer enables efficient and accurate PRM data construction without requiring human annotation or additional LLM calls. Using FoVer, we create PRM training data from formal logic and theorem proving tasks. Experiments on 12 reasoning benchmarks show that fine-tuning on our training data improves PRMs not only on math and logic reasoning tasks, which are informal variants of the training tasks, but also on NLI and BBH benchmarks, which differ substantially from the tasks used to construct the training data. These results demonstrate the practical effectiveness of FoVer, showing that PRM training data created using formal verification improves PRMs on informal reasoning tasks written in natural language. The datasets, models, and code are provided at https://github.com/psunlpgroup/FoVer.

1 Introduction

Process Reward Models (PRMs) have recently emerged as a promising approach for improving the reasoning capabilities of LLMs. They provide fine-grained process supervision over intermediate reasoning steps during training and inference (Uesato et al., 2022; Lightman et al., 2024). In practice, PRMs are usually created by fine-tuning LLMs to classify the correctness of individual reasoning steps, using training datasets annotated with step-level error labels on LLM-generated reasoning traces (Wang et al., 2024; Zhang et al., 2025).

Despite their growing adoption, creating PRM training data remains costly and noisy. Early studies (Uesato et al., 2022; Lightman et al., 2024) create PRM training data via human annotation, which is particularly expensive for obtaining step-level error labels and suffers from low inter-annotator agreement (Zheng et al., 2025). Although Monte Carlo roll-outs (Wang et al., 2024; Luo et al., 2024), also known as Math-Shepherd, provide an annotation-free alternative for training data creation, it requires multiple LLM calls to label each step and generates noisy training labels.

To address this gap, we propose FoVer, a novel framework for efficiently creating accurate PRM training data. As shown in Figure 1, FoVer leverages formal verification tools to assign accurate step-level error labels to reasoning traces for formal reasoning tasks such as logic and theorem proving, without relying on human annotation or repeated LLM calls. Using FoVer, we create PRM training data named FoVer-40K by annotating step-level error labels on LLM responses to formal logic and theorem proving tasks using Z3 (de Moura and Bjørner, 2008) and Isabelle (Nipkow et al., 2002). We then fine-tune Llama 3.1 8B (Llama Team, 2024) and Qwen 2.5 7B (Qwen Team, 2024) on FoVer-40K to use them as PRMs, which we refer to as FoVer-PRMs.

Although FoVer creates PRM training data from formal reasoning tasks, PRMs are typically used to detect mistakes in informal reasoning tasks written in natural language. Therefore, in our experiments, we evaluate PRMs trained with FoVer on widely used informal reasoning benchmarks. First, we evaluate FoVer-PRMs on informal logic and mathematical reasoning benchmarks. These benchmarks can be regarded as informal variants of the

Findings of the Association for Computational Linguistics: ACL 2026, pages 8246–8265
July 2-7, 2026

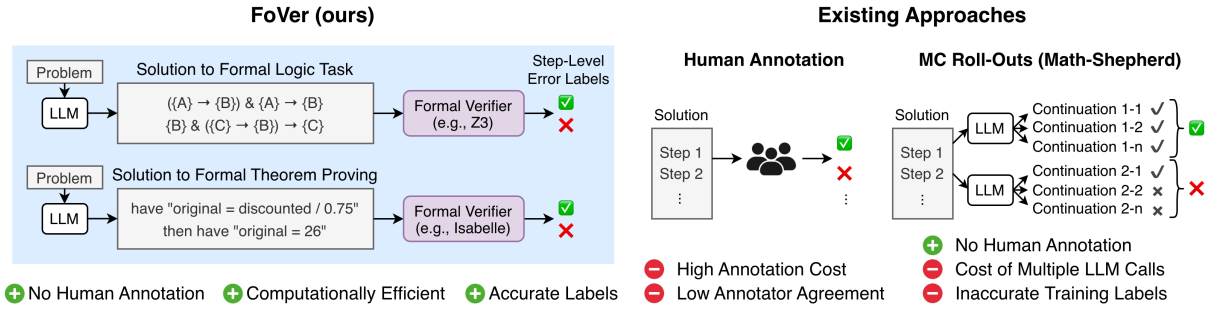
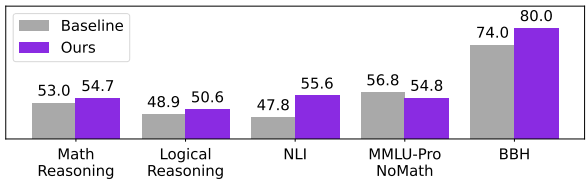
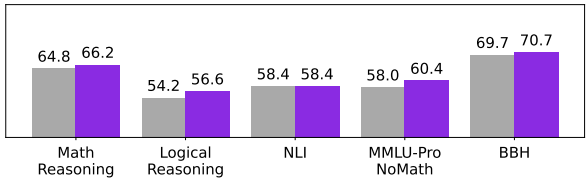


Figure 1: Comparison of FoVer with existing methods for PRM training data creation. FoVer efficiently produces accurate PRM training data for formal reasoning tasks by leveraging formal verification tools. In contrast, existing methods are costly and produce noisy labels, as they rely on human annotation or repeated LLM calls.



(a) PRMs based on Llama 3.1 8B



(b) PRMs based on Qwen 2.5 7B

Figure 2: Best-of-7 results across 5 categories of 12 reasoning benchmarks. PRMs trained with FoVer outperform baseline PRMs on average, showing that our training data created from formal reasoning improves PRMs on widely used reasoning benchmarks. Details are in Section 4.

training tasks, as FoVer-40K consists of formal logic and theorem proving tasks. We conduct evaluations on 6 widely used benchmarks, including LogicNLI (Tian et al., 2021), AQuA-RAT (Ling et al., 2017), and AIME (Art of Problem Solving, 2025). Using Best-of-K (Cobbe et al., 2021; Li et al., 2023), a standard evaluation approach for PRMs (Lightman et al., 2024; Wang et al., 2024), we show that FoVer-PRMs outperform baseline PRMs based on LLMs without additional fine-tuning. Second, we evaluate FoVer-PRMs on informal reasoning benchmarks involving tasks that substantially differ from those used to construct FoVer-40K. We use six benchmarks, including HANS (McCoy et al., 2019), MMLU (Paech, 2024), and BBH (Suzgun et al., 2023). The results show that FoVer-PRMs outperform baseline PRMs and achieve performance competitive with existing PRMs (e.g., Xiong et al., 2024; Zhang et al., 2025), whose training data is constructed via human annotation or Monte Carlo roll-outs that require substantially higher cost than FoVer.

Together, as shown in Figure 2, our experiments demonstrate the practical effectiveness of FoVer in improving PRMs on widely used informal reasoning benchmarks. These results showcase formal-to-informal transfer and cross-task generalization in PRM training using formally verified labels annotated on formal reasoning tasks.

Our main contributions are as follows:

- We propose FoVer, a method that efficiently constructs accurate PRM training data from formal reasoning tasks using formal verification, addressing the annotation cost and labeling inaccuracy of existing approaches.
- Our experiments show the practical effectiveness of FoVer, with PRM training data from formal reasoning tasks improving performance on informal reasoning tasks written in natural language, demonstrating formal-to-informal transfer and cross-task generalization in PRM training.

2 Background and Related Work

PRMs for reasoning tasks are typically created by fine-tuning LLMs on the classification task to detect errors in reasoning traces at the step level (Lightman et al., 2024; Wang et al., 2024). Training data for PRMs are often constructed by annotating step-level error labels on reasoning traces generated by LLMs. Given a problem $p \in \mathcal{P}$ from a reasoning task, an LLM generates a step-by-step solution consisting of k steps: $s = [s_1, s_2, \ldots, s_k] \sim \mathrm{LLM}(p)$. We then assign step-level error labels $[y_{s_1}, y_{s_2}, \ldots, y_{s_k}] \in \{0, 1\}^k$, which serves as the

supervision signal for PRM training. Here, we consider binary labels of correct and incorrect.

Despite extensive research on PRM training data construction, obtaining reliable step-level labels remains challenging. Existing approaches are costly and noisy, as they often rely on human annotation or computationally expensive automatic methods.

Human annotation. Human annotation (Uesato et al., 2022; Lightman et al., 2024) is the primary approach used in early studies. However, for step-level reasoning tasks, it is particularly costly, and achieving high inter-annotator agreement is difficult. For example, Zheng et al. (2025) discard 30% of the annotated solutions due to low agreement.

Monte Carlo roll-outs. Monte Carlo roll-outs (Wang et al., 2024; Luo et al., 2024), also known as Math-Shepherd, is a widely used method for automating PRM training data creation. MC roll-outs generate multiple continuations from a target reasoning step and use the frequency of reaching the correct final answer as an estimate of the correctness of that step.

Specifically, to label a step s_r in a solution $s = [s_1, \ldots, s_r, \ldots, s_k]$, the method samples multiple continuations from an LLM conditioned on the problem p and the prefix $s_{1:r}$:

$$
\begin{aligned}
[s_1, \ldots, s_r, c^{(1)}_{r+1}, \ldots, c^{(1)}_{K_1}] &\sim \mathrm{LLM}(\cdot \mid p, s_{1:r}) \\
[s_1, \ldots, s_r, c^{(2)}_{r+1}, \ldots, c^{(2)}_{K_2}] &\sim \mathrm{LLM}(\cdot \mid p, s_{1:r}) \\
\ldots [s_1, \ldots, s_r, c^{(n)}_{r+1}, \ldots, c^{(n)}_{K_n}] &\sim \mathrm{LLM}(\cdot \mid p, s_{1:r})
\end{aligned}
$$

Let a_i denote the final answer produced by the i-th continuation and let a^* be the ground-truth final answer. The quality of step s_r is estimated as $q_{s_r} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(a_i = a^*)$, which can be used as a soft label or binarized to obtain a hard label.

Although widely adopted, MC roll-outs suffer from high computational cost due to the need to generate multiple continuations per step, and the estimated labels can be noisy and inaccurate.

Other approaches. Perturbations (Yang et al., 2022; Ma et al., 2023; Paul et al., 2024) is another automated approach that introduces mistakes into generated solutions by applying heuristic methods or using LLMs. However, it results in training data with artificial and unnatural errors. Some work creates PRM training data by annotating error labels using stronger LLMs (Zhang et al., 2025; Zeng et al., 2025), but this approach results in merely distilling the capabilities of stronger LLMs.

3 FoVer

We propose FoVer, a method for creating PRM training data by annotating step-level error labels on formal reasoning tasks using formal verification tools. Using FoVer, we create PRM training data named FoVer-40K from first-order logical reasoning tasks using Z3 and from formal theorem proving tasks using Isabelle.

3.1 Step-level Formal Verification

Background: Formal verification. Formal reasoning tasks (or symbolic reasoning tasks), including formal logic and formal theorem proving, are problems defined and solved using formal syntax and rules (Nawaz et al., 2019; Clark et al., 2021). Formal verification tools, such as automated solvers and theorem provers, are used to formally verify solutions to formal reasoning tasks (Zhou et al., 2024; Yang et al., 2025). The tools differ in the methods they use and in the groups of tasks to which they apply. For example, Z3 (de Moura and Bjørner, 2008) is an SMT solver, which extends SAT solvers with background theories, applicable to decidable subsets of first-order logic. Isabelle/HOL (Nipkow et al., 2002) is an interactive theorem prover applicable to higher-order logic.

To illustrate the process of formal verification, we present a simple example of a first-order logic reasoning (logical entailment) task that can be verified using a SAT solver:

$$\text{Context: } A \to B,\ B \to C,\ A \quad \text{Hypothesis: } C$$

Assume that the predicted answer is "entailment," meaning the hypothesis is logically entailed by the context. We want to verify whether this prediction is correct, which is equivalent to showing that the following implication is valid:

$$\big((A \to B) \land (B \to C) \land A\big) \ \to \ C.$$

We can check this with a SAT solver using resolution: combine the premises with the negation of the hypothesis and test its satisfiability in Conjunctive Normal Form (CNF):

$$(\neg A \lor B) \land (\neg B \lor C) \land A \land \neg C.$$

If the solver reports this formula unsatisfiable, then the entailment is confirmed. SAT solvers such as Z3 automate this process and thus can formally verify the correctness of the prediction.

Step-level formal verification. In this paper, we propose using formal verification tools to verify solutions at the step level. As illustrated above, these tools have primarily been applied to solution-level verification. Our key idea is that, for certain tasks, they can also be employed to verify the correctness of individual reasoning steps. For instance, for the above problem, suppose an LLM produces the following step-by-step solution:

$$\begin{aligned} &\text{Step 1:} \quad \big((A \to B) \wedge A\big) \;\to\; B. \\ &\text{Step 2:} \quad \big(B \wedge (A \to B)\big) \;\to\; C. \end{aligned}$$

It can be verified at the step level: we adopt a simple strategy that evaluates the logical correctness of each step independently. Once we check that each step only uses the provided context and preceding results, we can verify the logical correctness by testing the satisfiability of:

$$\begin{aligned} &\text{Step 1:} \quad (\neg A \vee B) \wedge A \wedge \neg B. \\ &\text{Step 2:} \quad B \wedge (\neg A \vee B) \wedge \neg C. \end{aligned}$$

The SAT solver will show that the first formula is unsatisfiable, indicating Step 1 is correct, and the second is satisfiable, indicating Step 2 is incorrect. This procedure provides step-level verification for the solution to the logical entailment task.

In this work, we implement step-level verification for formal logic (logical entailment) using Z3 and theorem-proving tasks using Isabelle to create PRM training data.

3.2 FoVer Framework

FoVer synthesizes PRM training data in two stages, as shown in Figure 3. In the first stage, given a problem $p \in \mathcal{P}$ from a formal reasoning task that is compatible with formal verification, an LLM generates a step-by-step formal solution consisting of k steps: $s = [s_1, s_2, \ldots, s_k] \sim \mathrm{LLM}(p)$. The solution s may contain logical errors but is required to follow the format compatible with a formal verification tool. In this work, we provide a few-shot demonstration to guide LLMs. Alternatively, it is also possible to use models trained for specific formal verification tools (Yang et al., 2023; Xin et al., 2024). LLMs may generate solutions in an invalid format, so we generate multiple solutions until we obtain one with a valid format.

In the second stage, the formal verification tool assigns step-level error labels $[y_{s_1}, y_{s_2}, \ldots, y_{s_k}] \in \{0,1\}^K$, without the need for human annotation:

$$\begin{aligned} &[y_{s_1}, y_{s_2}, \ldots, y_{s_k}] \\ &\quad = \mathrm{FormalVerification}(p, [s_1, s_2, \ldots, s_k]), \end{aligned}$$

where $\mathrm{FormalVerification}(,)$, a formal verification tool, accurately outputs label $y_{s_k} = 1$ if the solution step s_k is correct, $y_{s_k} = 0$ otherwise.

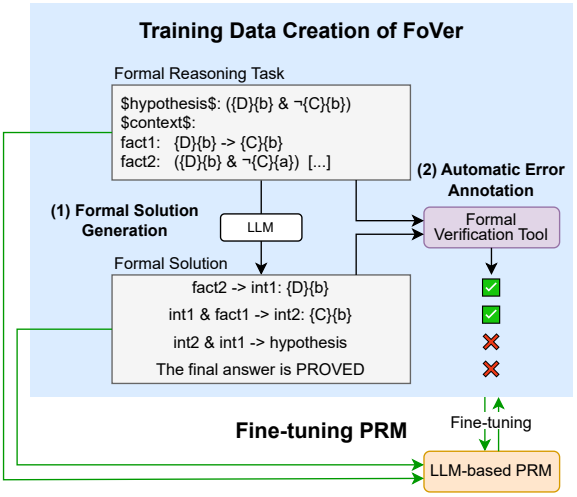


Figure 3: Top: FoVer creates PRM training data from formal reasoning tasks using formal verification tools. Bottom: Using the training data, we fine-tune PRMs on the task of step-level verification.

Using training data created with FoVer, we fine-tune an LLM on the step-level binary classification task to serve as a PRM ($\mathcal{P} \times \mathcal{S} \to [0,1]^K$), which is the standard approach for creating classification-based PRMs (Zhang et al., 2025). The cross-entropy training objective for PRMs is given by:

$$\mathcal{L} = \sum_{i=1}^{K} \left(y_{s_i} \log r_{s_i} + (1 - y_{s_i}) \log(1 - r_{s_i})\right),$$

where r_{s_i} is the step-level score for step s_i predicted by the PRM. In our implementation, we fine-tune LLM in a text generation task to output the token "correct" or "incorrect" for each step.

3.3 FoVer-40K Dataset

Using FoVer, we synthesize PRM training data that includes step-level binary error labels on the formal logic and formal theorem proving tasks, which we refer to as FoVer-40K. As shown in Table 1, FoVer-40K includes 40K steps in reasoning traces generated by Llama 3.1 8B (Llama Team, 2024) and Qwen 2.5 7B (Qwen Team, 2024) with error labels annotated by formal verification tools, Z3 and Isabelle. Figure 4 shows example data.

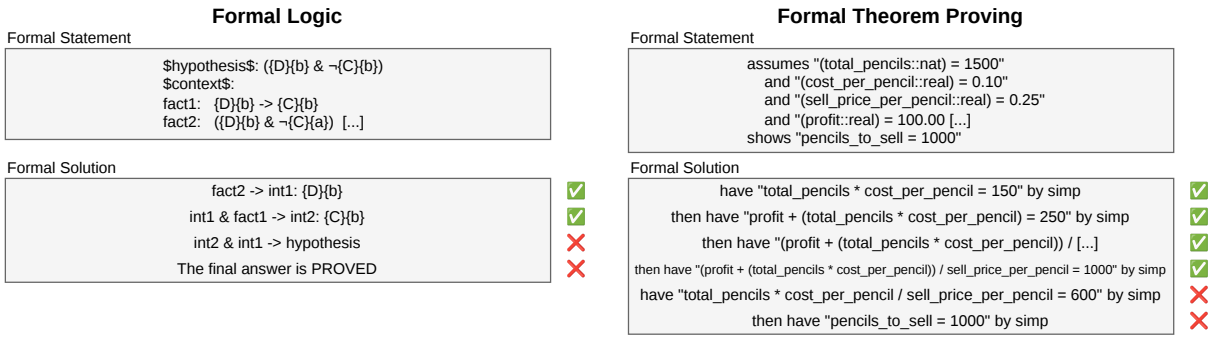


Figure 4: Examples from FoVer-40K.

Tasks	Source Datasets	Formal Verification Tools	Step-by-step Solutions	Error Labels	
				# of Steps	% Error
Formal Logic	FLDx2	Z3	Llama 3.1 8B	10,000	50%
			Qwen 2.5 7B	10,000	50%
Formal Theorem Proving	GSM8K, MetaMathQA, Big-Math	Isabelle	Qwen 2.5 7B	20,000	50%

Table 1: Statistics of the FoVer-40K training dataset.

The FoVer-40K dataset is created from FLDx2 (Morishita et al., 2023, 2024) and GSM8K-level mathematical reasoning tasks. We select these tasks based on diversity, simplicity, and generalization considerations. For the formal logic task, we selected FLDx2 because it offers the greatest diversity among those expressible via deduction rules (Morishita et al., 2024). For mathematical reasoning, we selected GSM8K-level datasets to simplify the verification pipeline.

Formal logic. We use the logical entailment task in FLDx2 (Morishita et al., 2023, 2024), a dataset for multi-step first-order logic deduction, in which the goal is to determine whether a hypothesis is entailed by a given set of premises. We generate step-by-step formal solutions with the LLMs and annotate step-level error labels using Z3.

Formal theorem proving. We use the task of formal theorem proving for verifying solutions to math word problems (Wu et al., 2022; Zhou et al., 2024). The conditions of a math word problem are presented as premises, a candidate final answer is formulated as a hypothesis, and the task is to generate a proof for this statement. We generate formal proofs with the LLMs in the task of verifying the solutions to GSM8K-level problems, including GSM8K (Cobbe et al., 2021), GSM8K-based cases in MetaMathQA (Yu et al., 2024), and math word problems in Big-Math (Albalak et al., 2025). We then annotate step-level error labels using Isabelle.

4 Experimental Results

This section evaluates our FoVer-PRMs fine-tuned on FoVer-40K. While FoVer-40K includes step-level error labels on formal reasoning tasks, PRMs are often used to detect mistakes in informal reasoning tasks written in natural language. Therefore, we evaluate FoVer-PRMs on widely used informal reasoning benchmarks to answer the following research questions.

- RQ1: Does training on FoVer-40K improve PRMs on informal logic and math reasoning tasks, which are informal variants of the training tasks? (formal-to-informal transfer, §4.2)
- RQ2: Does training on FoVer-40K improve PRMs on informal reasoning tasks that largely differ from the training tasks, such as NLI and BBH? (cross-task generalization, §4.3)

4.1 Experimental Setting

Evaluation methodology. We evaluate PRMs using the Best-of-K performance on reasoning benchmarks and step-level verification performance, both of which are widely used approaches to assess PRMs (Li et al., 2023; Zhang et al., 2024b). First, **Best-of-K** (Cobbe et al., 2021; Li et al., 2023) uses PRMs to select the best solution from multiple candidates generated by LLMs for the same input. The performance of the selected solutions indirectly indicates the verification performance of the PRMs. We use few-shot demonstrations to generate $K = 7$ step-by-step solutions from Llama 3.1 8B and Qwen 2.5 7B with a temperature of 0.5. Following prior work (Lightman et al., 2024; Wang et al., 2024), we then use PRMs to score each step, and the solution score is the minimum score across all steps. The solution with the highest solution score is selected. For large

PRMs		Logic		Average	Math				Average
		FOLIO	LogicNLI		GSM8K	MATH	AQuA	AIME	
PRMs based on Llama 3.1 8B	Llama 3.1 8B	58.6	39.2	48.9	87.6	**54.4**	66.1	4.0	53.0
	FoVer-Llama3.1-8B	**60.1**	**41.2**	**50.6**	**88.4**	53.6	**71.7***	**5.2**	**54.7***
PRMs based on Qwen 2.5 7B	Qwen 2.5 7B	**64.0**	44.4	54.2	90.0	76.4	80.3	**12.4**	64.8
	FoVer-Qwen2.5-7B	63.5	**49.6**	**56.6**	**92.4**	**79.2**	**81.1**	12.0	**66.2***

Table 2: Best-of-K (K=7) performance on informal logic and math benchmarks. The first rows represent the baseline PRMs based on LLMs without additional training. FoVer-PRMs exhibit significantly better performance, showing that FoVer-40K improves verification on informal logic and math reasoning. *: Statistically significant improvement over the baseline PRMs in the first row ($p < 0.05$, paired bootstrap).

datasets, we use 250 randomly sampled examples from each dataset. Second, we evaluate **step-level verification** performance on math reasoning tasks in ProcessBench (Zheng et al., 2025), which includes human annotated binary step-level error labels. ProcessBench includes labels only for the earliest error in each reasoning traces, so we use the steps up to the first error in each solution.

Evaluated PRMs. We evaluate classification-based PRMs that use Llama 3.1 8B (Llama Team, 2024) and Qwen 2.5 7B (Qwen Team, 2024) as the backbone. We choose these LLMs because they are widely used as backbones for PRMs, which allows for direct comparison with PRMs introduced in prior work. Our PRMs, referred to as **FoVer-PRMs**, are obtained by fine-tuning these LLMs on FoVer-40K. **Baseline PRMs:** We compare with baselines that use the backbone LLMs as PRMs without additional fine-tuning. **Existing PRMs:** In Section 4.3, we also evaluate five PRMs introduced in prior work (Table 4). Among PRMs based on Llama 3.1 8B, we evaluate RLHFlow-Llama3.1-8B trained on the DeepSeek or Mistral data (Xiong et al., 2024), which include error labels on solutions generated by stronger models on GSM8K and MATH acquired via Monte Carlo roll-outs (Wang et al., 2024). Among PRMs based on Qwen 2.5 7B, we evaluate Qwen2.5-Math-7B-PRM800K (Zheng et al., 2025), which is trained on human-annotated labels on MATH, and Qwen2.5-Math-PRM-7B (Zhang et al., 2025), which is trained on labels synthesized using Monte Carlo roll-outs and verification by a stronger model. We also evaluate Qwen2.5-7B-Skywork-PRM (He et al., 2024), which is trained on math and coding.

4.2 Results on Logic and Math

First, we evaluate FoVer-PRMs on widely-used informal logic and math reasoning benchmarks, which are informal variants of formal logic and theorem proving tasks included in FoVer-40K.

PRMs	GSM8K	MATH	Olympiad	Omni	Ave.
Llama 3.1 8B	70.0	67.4	68.3	63.9	67.4
FoVer-Llama3.1-8B-PRM	**81.2***	**76.4***	**75.5***	**76.3***	**77.3***
Qwen 2.5 7B	75.5	78.2	76.2	73.6	75.9
FoVer-Qwen2.5-7B-PRM	**86.6***	**88.0***	**84.0***	**84.5***	**85.8***

Table 3: Step-level binary classification performance of PRMs on math reasoning tasks in ProcessBench (AUROC). FoVer-PRMs exhibit significantly better performance than the baselines, showing that FoVer-40K improves verification on informal math reasoning. *: Statistically significant improvement over the baseline PRMs in the first rows ($p < 0.05$, paired bootstrap).

Results of Best-of-K. We evaluate the Best-of-K performance of PRMs on two logical reasoning benchmarks: FOLIO (Han et al., 2024), LogicNLI (Tian et al., 2021), and four math benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AQuA-RAT (Ling et al., 2017), AIME (2016-2024) (Art of Problem Solving, 2025). Table 2 reports the Best-of-K performance, showing that FoVer-PRMs outperform the baseline PRMs on average both on the informal logic and math tasks.

Results on ProcessBench. We also evaluate the PRMs on the step-level binary classification (error detection) task on mathematical reasoning tasks in ProcessBench (Zheng et al., 2025). ProcessBench includes human-annotated step-level error labels (correct vs. incorrect) for reasoning traces on four math benchmarks: GSM8K, MATH, OlympiadBench, and Omni-Math. Table 3 shows the step-level verification performance of PRMs, measured in AUROC. The results show that fine-tuning on FoVer-40K significantly improves PRM performance at detecting step-level mistakes from the baseline PRMs in informal math reasoning tasks.

PRM	Tasks in Training Data	Step-Level Error Annotation	# LLM Calls per Solution for Labeling	Does Not Require Stronger LLMs	No Human Annotation	Accurate Annotation
RLHFlow-Llama3.1-8B (M, D)	Math Reasoning	MC Roll-out	8 × # Steps	✓	✓	✗
Qwen2.5-Math-7B-PRM800K	Math Reasoning	Human Annotation	1	✓	✗	✗
Qwen2.5-7B-Skywork-PRM	Math Reasoning, Coding	MC Roll-out	(not reported)	✓	✓	✗
Qwen2.5-Math-PRM-7B	Math Reasoning	MC Roll-out, LLMs	8 × # Steps	✗	✓	✗
FoVer-PRM (Ours)	Formal Logic, Formal Theorem Proving	Formal Verification	1	✓	✓	✓

Table 4: Training data creation methods for PRMs evaluated in Section 4.3.

	PRMs	NLI		Average	MMLU	BBH			Average
		ANLI	HANS		Pro-NoMath	Temporal	Tracking	Sorting	
PRMs based on Llama 3.1 8B	Llama 3.1 8B	26.4	69.2	47.8	56.8	90.4	90.0	41.6	74.0
	RLHFlow-Llama3.1-8B-M	26.8	77.6*	52.2*	56.8	92.0	90.8	39.2	74.0
	RLHFlow-Llama3.1-8B-D	27.2	74.4*	50.8*	**57.6**	**98.8***	92.8	40.0	77.2*
	FoVer-Llama3.1-8B-PRM (ours)	**30.0**	**81.2***	**55.6***	54.8	97.2*	**96.0***	**46.8***	**80.0***
PRMs based on Qwen 2.5 7B	Qwen 2.5 7B	**32.4**	84.4	**58.4**	58.0	91.2	89.2	28.8	69.7
	Qwen2.5-Math-7B-PRM800K	29.6	83.2	56.4	**62.0***	82.0	90.8	28.4	67.1
	Qwen2.5-7B-Skywork-PRM	29.2	81.2	55.2	58.8	84.0	91.2	**31.6**	68.9
	Qwen2.5-Math-PRM-7B	27.6	**86.0**	56.8	60.4	86.4	**92.8***	28.8	69.3
	FoVer-Qwen2.5-7B-PRM (ours)	30.8	**86.0**	**58.4**	60.4	**92.4**	90.0	29.6	**70.7**

Table 5: Best-of-K (K=7) performance on tasks that differ from tasks in training data of FoVer-PRMs and existing PRMs (i.e., unseen tasks). The first rows represent the baseline PRMs based on LLMs without additional training. The next set of rows contains existing PRMs introduced in prior work. The last rows show our FoVer-PRMs. *: Statistically significant improvement over the baseline PRMs in the first row ($p < 0.05$, paired bootstrap).

Answer to RQ1: Formal-to-Informal Transfer

PRMs trained on FoVer-40K outperform baseline PRMs on average on informal math and logic benchmarks. These results demonstrate that PRM training on formally verified error labels on formal reasoning tasks exhibits formal-to-informal transfer and improves performance on informal variants of the formal training tasks.

4.3 Results on Unseen Tasks

Second, we evaluate FoVer-PRMs on reasoning tasks that are substantially different from the formal logic and theorem proving tasks in FoVer-40K, which we refer to as unseen tasks. We conduct evaluations on 6 reasoning benchmarks spanning three tasks: NLI, including ANLI (Nie et al., 2020) and HANS (McCoy et al., 2019); MMLU-Pro-NoMath (Paech, 2024); and BIG-Bench Hard, including temporal sequences, tracking shuffled objects (three objects), and word sorting (Suzgun et al., 2023). Here, in addition to FoVer-PRMs, we evaluate existing PRMs introduced in prior work (Table 4). Since these PRMs are trained on mathematical reasoning or coding tasks, the reasoning benchmarks evaluated in this section are also unseen for the existing PRMs.

Table 5 shows the Best-of-K performance of PRMs on the unseen tasks. First, compared to the baseline PRMs, FoVer-PRMs improve performance on average on almost all tasks. Furthermore, FoVer-PRMs are competitive with existing PRMs, whose training datasets are created at substantially higher cost. In particular, FoVer-Llama3.1-8B-PRM achieves the best performance on the majority of the benchmarks, compared with existing PRMs based on the same LLM.

Answer to RQ2: Cross-Task Generalization

PRMs trained on formal reasoning tasks in FoVer-40K exhibit cross-task generalization, improving performance on informal reasoning tasks such as NLI and BBH that are substantially different from the formal logic and theorem proving tasks in FoVer-40K.

5 Analysis

Ablation on training tasks. FoVer-40K consists of formal logic and theorem proving tasks. To assess the contribution of each task, we train PRMs on each task separately and evaluate their performance. Table 6 reports the Best-of-K performance of PRMs based on Llama 3.1 8B trained

Formal Logic	Formal Theorem	GSM8K	LogicNLI	HANS	Temporal	Tracking	Ave.
–	–	87.6	39.2	69.2	90.4	90.0	75.3
✓	–	86.8	**46.0***	75.2*	96.8*	92.4	79.4*
–	✓	**90.4**	44.0	76.8*	96.0*	94.8*	80.4*
✓	✓	88.4	41.2	**81.2***	**97.2***	**96.0***	**80.8***

Table 6: Ablation on training tasks for Best-of-K (K=7) performance of PRMs based on Llama 3.1 8B trained on different subsets of FoVer-40K. The top row represents the baseline PRM, and the last row represents the PRM trained on the original FoVer-40K.

PRMs	GSM8K	LogicNLI	HANS	Temporal	Tracking	Ave.
Qwen 3 4B	**93.2**	66.4	**89.6**	87.6	**95.6**	86.5
FoVer-Qwen3-4B	91.6	**70.0**	**89.6**	**89.2**	**95.6**	**87.2**

Table 7: Best-of-K (K=7) performance of PRMs based on Qwen 3 4B. Fine-tuning on FoVer-40K (second row) yields improved or comparable performance on all but one benchmark.

on different subsets of FoVer-40K. The results show that each task is independently effective and leads to significant improvements over the baseline PRMs on average. Looking more closely, we observe that combining the two tasks during training improves performance on unseen tasks (HANS, Temporal, and Tracking), while PRMs trained on individual tasks achieve better performance on their corresponding logic and math benchmarks. In light of this observation, when creating PRMs intended for use in a specific domain with available in-distribution training data, it may be preferable not to include additional domains in the training data. We will add this discussion to the updated version of the paper.

Robustness to stronger backbone LLM. To evaluate the robustness of PRM training with FoVer, we additionally evaluate PRMs based on Qwen 3 4B (Qwen Team, 2025), which is newer and stronger than Llama 3.1 8B and Qwen 2.5 7B. Table 7 reports the Best-of-K performance of PRMs based on Qwen 3 4B with and without fine-tuning on FoVer-40K. The results show that fine-tuning on FoVer-40K yields improved or comparable performance across nearly all benchmarks, aside from GSM8K. These findings indicate that FoVer-40K, which is constructed from responses generated by weaker models, improves PRMs built on stronger LLM backbones, demonstrating the robustness of training with FoVer with respect to backbone LLM choice.

Ablation on training data size. Figure 5 presents

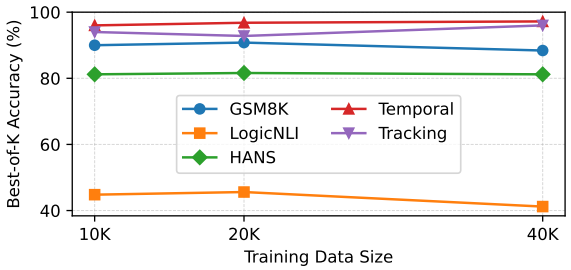


Figure 5: Ablation on training data size for PRMs based on Llama 3.1 8B trained on randomly sampled subsets of FoVer-40K with 10K and 20K instances.

Problem: Today, Sarah went to the art show. Between what times could they have gone? [...] Lisa saw Sarah working at the office from 11am to 2pm. [...] Options: (A) 11am to 2pm (B) 7pm to 9pm [...]
Ground truth answer: (B) 7pm to 9pm

Step	Solution from Llama 3.1 8B	Baseline	FoVer
2	Lisa saw Sarah working at the office from 11am to 2pm, so Sarah could have gone to the art show from 2pm.	0.98	0.99
7	From the options, we can see that 11am to 2pm is the only option that fits this time frame.	**0.94**	**0.06**

(a) Positive example of FoVer-PRM on the temporal sequence task. FoVer-Llama3.1-8B-PRM detects an error in Step 7, whereas the baseline erroneously assigns a high score.

Problem: For what integer x is $\frac{3}{5} < \frac{x}{7} < \frac{7}{9}$?

Step	Solution from Llama 3.1 8B	Baseline	FoVer
0	We can rewrite the inequality as $\frac{3}{5} < \frac{x}{7} < \frac{7}{9}$.	1.00	1.00
1	Multiplying the inequality by 315 to eliminate the fractions, we obtain $189 < 45x < 245$.	1.00	0.98
2	Dividing the inequality by 45, we get $\frac{189}{45} < x < \frac{245}{45} \Rightarrow 4.2 < x < 5.44$.	**1.00**	**0.68**

(b) Negative example for FoVer-PRM on MATH. FoVer-Llama3.1-8B-PRM is too strict in Step 2 and considers rounding $5.44444\ldots$ to 5.44 as a mistake.

Table 8: Example outputs from FoVer-PRMs.

an ablation study on training data size for the Best-of-K performance of PRMs based on Llama 3.1 8B trained on randomly sampled subsets of FoVer-40K with 10K and 20K instances. The results show that PRMs trained on a subset with 10K instances already achieve performance comparable to those trained on the full dataset, showing the data efficiency in the FoVer training.

Case study. Table 8 shows example outputs from FoVer-PRM and baseline PRMs. Table 8a shows a **positive example** on a solution to the temporal

sequence task in BBH. The task is to find a free time slot that does not overlap with any events in a given list of past time intervals, which is distinct from formal reasoning tasks in FoVer-40K. This example shows that FoVer-Llama3.1-8B-PRM properly understands the task and solutions described in natural language and correctly detects a logical error in Step 7, showing the formal-to-informal transfer and cross-task generalization in PRM training with FoVer. Table 8b shows a **negative example** illustrating a potential side effect of FoVer. In this example, FoVer-Llama3.1-8B-PRM assigns a low score when a solution step rounds $\frac{245}{45} = 5.44444\ldots$ to 5.44 for conciseness. While this can be considered a mistake under strict evaluation, it is often acceptable when applied to intermediate values where the exact precision is not required for the correctness of the final result. Training on formally annotated labels in FoVer may cause PRMs to become overly strict, assigning lower scores to intermediate steps that are acceptable but not strictly precise.

6 Discussion

Our experiments show that PRMs trained on symbolic solutions for formal tasks improve performance on widely used informal reasoning tasks, demonstrating effective formal-to-informal transfer in PRM training. Compared to prior work that directly fine-tunes LLMs on symbolic reasoning traces, our results indicate that PRM training yields stronger transferability. For instance, Morishita et al. (2024) train LLMs on reasoning traces from a formal logic dataset to improve general reasoning, but report limited transfer in a naive setup. To address this issue, they convert symbolic solutions into natural language and apply RecAdam (Chen et al., 2020). In contrast, our results show that PRM training exhibits formal-to-informal transfer without these modifications. We attribute this advantage to the consistency of the output space in PRM training. Regardless of the input format, PRMs generate a binary label indicating whether each step is correct or incorrect, which reduces shifts in model behavior that can arise when LLMs are directly fine-tuned to generate symbolic reasoning traces.

7 Conclusion

We introduce FoVer, a method for constructing PRM training data from formal reasoning tasks by annotating step-level error labels using formal verification tools. In contrast to existing approaches that are costly and noisy, FoVer efficiently produces accurate PRM training data without relying on human annotation or repeated LLM calls. Experimental results show that PRMs trained on data generated by FoVer achieve improved performance across reasoning benchmarks in math, logic, NLI, and BBH, demonstrating informal-to-formal transfer and cross-task generalization in PRM training. These findings show that formal verification provides a practical approach to PRM training data construction, improving PRMs on widely used informal reasoning tasks written in natural language.

Limitations

This work focuses on classification-based PRMs that generate a single score to each step. Although recent studies introduce PRMs that provide step-level reasoning for error classification (Khalifa et al., 2025; Feng et al., 2025; Kim et al., 2025), we leave the study of this type of PRM to future work, as they are not directly comparable with classification-based PRMs. PRMs with step-level reasoning incur substantially higher computational costs for verification, often multiple times greater than those required to generate the target reasoning traces, which limits their practical usefulness.

Acknowledgments

This work was supported by NSF IIS-2338418 and DMS-2533995. We thank Jin Peng Zhou for providing guidance on the use of his code (Zhou et al., 2024). We also thank Terufumi Morishita for the valuable discussions and for his assistance with the FLDx2 dataset (Morishita et al., 2024), and we appreciate NLP Colloquium JP for providing the opportunity to connect with him. We thank Hieu N. Nguyen for his suggestions on improving the presentation of this work.

References

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*.

Art of Problem Solving. 2025. Aime problems and solutions.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn:

Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

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. *arXiv preprint arXiv:2110.14168*.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: an efficient smt solver. In *2008 Tools and Algorithms for Construction and Analysis of Systems*, pages 337–340. Springer, Berlin, Heidelberg.

DeepSeek-AI. 2025. Deepseek-r1 incentivizes reasoning capability in large language models. *Nature*.

Zhangyin Feng, Qianglong Chen, Ning Lu, Yongqian Li, Siqi Cheng, Shuangmu Peng, Duyu Tang, Shengcai Liu, and Zhirui Zhang. 2025. Is PRM necessary? problem-solving RL implicitly induces PRM capability in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024. Skywork-o1 open series. `https://huggingface.co/Skywork`.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.

Seungone Kim, Ian Wu, Jinu Lee, Xiang Yue, Seongyun Lee, Mingyeong Moon, Kiril Gashteovski, Carolin Lawrence, Julia Hockenmaier, Graham Neubig, and Sean Welleck. 2025. Scaling evaluation-time compute with reasoning models as process evaluators. *arXiv preprint arXiv:2503.19877*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *Computer Aided Verification*, pages 1–35, Berlin, Heidelberg. Springer Berlin Heidelberg.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

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*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Llama Team. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in

language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2023. Learning deductive reasoning from synthetic corpus based on formal logic. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25254–25274. PMLR.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of LLMs via principled synthetic logic corpus. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, page 625–635, Berlin, Heidelberg. Springer-Verlag.

M. Saqib Nawaz, Moin Malik, Yi Li, Meng Sun, and M. Ikram Ullah Lali. 2019. A survey on theorem provers in formal methods. *arXiv preprint arXiv:1912.03028*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. 2002. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Sam Paech. 2024. Mmlu-pro-nomath.

Sarah Pan, Vladislav Lialin, Sherin Muckatira, and Anna Rumshisky. 2023. Let's reinforce step by step. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta. Association for Computational Linguistics.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.

Stephan Schulz. 2002. E - a brainiac theorem prover. *AI Commun.*, 15(2,3):111–126.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

The Coq Development Team. 2024. The Coq reference manual – release 8.19.0. https://coq.inria.fr/doc/V8.19.0/refman.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Sean Welleck. 2023. Neural theorem proving tutorial. https://github.com/wellecks/ntptutorial.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 32353–32368. Curran Associates, Inc.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*.

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. An implementation of generative prm. https://github.com/RLHFlow/RLHF-Reward-Modeling.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 89–105, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin E. Lauter, Swarat Chaudhuri, and Dawn Song. 2025. Position: Formal mathematical reasoning—a new frontier in AI. In *Forty-second International Conference on Machine Learning Position Paper Track*.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21573–21612. Curran Associates, Inc.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2025. Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. 2025. VersaPRM: Multi-domain process reward model via synthetic reasoning data. In *Forty-second International Conference on Machine Learning*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024b. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

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

Vienna, Austria. Association for Computational Linguistics.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024. Don't trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*.

Table of Contents of Appendix

A Additional Related Work

Applications of PRMs. PRMs can be used to supervise LLM reasoning during training and inference. For **training**, PRMs can generate reward signals, particularly in reinforcement learning settings (Pan et al., 2023; Zhang et al., 2024a). They can be applied either to re-rank candidate responses from the policy or to provide direct reward (Uesato et al., 2022). For **inference**, PRMs can guide response selection and refinement through Best-of-K (Li et al., 2023), self-correction (Saunders et al., 2022; Madaan et al., 2023), and step-level search (Ma et al., 2023; Snell et al., 2025).

Outcome-based training for PRMs. Provided the difficulty of collecting step-level error labels, recent work proposes methods to train PRMs without using step-level labels. Yuan et al. (2025) proposes implicit PRMs that can be trained only using final answers. They theoretically show that ORMs trained with the reward that is parameterized by the log-likelihood ratio of two causal language models (e.g., DPO (Rafailov et al., 2023)) implicitly learns a Q function and can be used as PRMs. In addition, recent work (Feng et al., 2025; Kim et al., 2025) reports that existing Large Reasoning Models like DeepSeek-R1 (DeepSeek-AI, 2025) have strong process-level rewarding capabilities on

mathematical reasoning tasks, while they are not explicitly trained for process-level rewarding.

Formal logic and theorem proving. Formal logic tasks have been used to evaluate and improve the reasoning of LLMs (Clark et al., 2021; Tafjord et al., 2021; Morishita et al., 2024). These tasks are often formulated in logical languages compatible with automated solvers such as Z3 (de Moura and Bjørner, 2008), Vampire (Kovács and Voronkov, 2013), or E (Schulz, 2002). Separately, formal theorem proving has been studied in the context of LLM-based proof generation (Polu and Sutskever, 2020; Yang et al., 2023; Xin et al., 2024), where proof assistants such as Isabelle/HOL (Nipkow et al., 2002), Coq (The Coq Development Team, 2024), and Lean (Moura and Ullrich, 2021) are used to validate generated proofs. In contrast, our work presents the first attempt to use verification outputs from these tools as training signals for reward models. Moreover, while these tools are designed and have been used for solution-level verification, we introduce a novel use of them to provide step-level supervision for PRMs.

B Examples from FoVer-40K

We provide examples from FoVer-40K.

Formal logic. Here is an example of the data in the formal logic task based on the initial responses from Llama 3.1 8B.

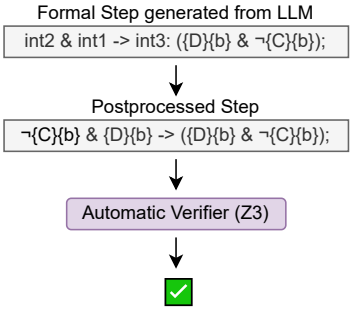
```
{
  'id': 'fldx2-train-020948_Llama-3.1-8B-Instruct',
  'error_labels': [True, False],
  'messages': [
  {'role': 'user', 'content': '** Problem **
    Based on the provided facts ($context$), either prove or
    ↪  disprove the hypothesis or state that it is unknown.
    ↪  The facts and the hypothesis are written in logical
    ↪  formulas as follows: capital letters such as "{A}",
    ↪  "{B}", "{AB}" are predicates, small letters such as
    ↪  "{a}", "{b}", "{ab}" are constants, "&" is logical
    ↪  conjunction, "v" is logical disjunction, "¬" is
    ↪  negation, "->" is implication, "(x)" is "for all x",
    ↪  and "(Ex)" is "for some x".

    $hypothesis$: ¬{B}

    $context$:\nfact1: ¬(¬{F} & ¬{G}) -> ¬{E}
    fact2: ¬{CF}\nfact3: ¬{IG}
    fact4: ¬{DQ}
    fact5: ¬{CN}\nfact6: ¬{BA}
    fact7: {B} -> (¬{DC} & ¬{A})
    fact8: ¬{DM}\nfact9: {C} -> {B}
    fact10: ¬{IF}
    fact11: ¬{JJ}
    fact12: (¬{GL} & ¬{AP})
    fact13: ¬{E} -> ({D} & {C})
    fact14: (¬{A} & ¬{B})
    fact15: (¬{DC} & ¬{L})
    fact16: ¬{IJ}
    fact17: ¬{EM}
    fact18: {H} -> ¬(¬{F} & ¬{G})
    fact19: ¬{DO}

    ** Task **
    Your task is to evaluate the accuracy of each step in the
    ↪  provided solution to the above question. For each
    ↪  step, respond with "correct" if the reasoning is
    ↪  logically valid and mathematically sound, or if the
    ↪  step is a general statement or transition that does
    ↪  not contain reasoning. Respond with "incorrect" if
    ↪  the step includes any errors or flawed logic.

    ** Sotluion **
    fact7 -> int1: {B} -> (¬{DC} & ¬{A})'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'fact6 -> int2: ¬{B}'},
    {'role': 'assistant', 'content': 'incorrect'}
  ],
}
```

Formal theorem proving. Here is an example of the data in the formal proof task based on the initial responses from Qwen 2.5 7B.

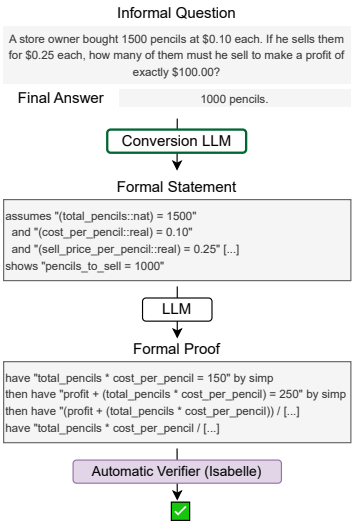
```
{
  'id':
  ↪  'metamathqa_GSM_Rephrased_076554_Qwen2.5-7B-Instruct',
  'error_labels': [True, True, True, True, True, True,
  ↪  False],
  'messages': [
    {'role': 'user', 'content': '** Problem **
    Generate a proof for the following theorem in the
    ↪  Isabelle proof assistant format.

    theorem example:
        assumes "(Jorge_age_2005::nat) = 16"
        and "(Jorge_age_difference::nat) = 24"
        and "(Jorge_current_age::nat) = Jorge_age_2005 +
        ↪  (2022 - 2005)"
        and "(Simon_current_age::nat) = Jorge_current_age +
        ↪  Jorge_age_difference"
        and "(Simon_age_2010::nat) = Simon_current_age +
        ↪  (2010 - 2022 + 2005)"
        shows "Simon_age_2010 = 50"

    ** Task **\nYour task is to evaluate the accuracy of each
    ↪  step in the provided solution to the above question.
    ↪  For each step, respond with "correct" if the
    ↪  reasoning is logically valid and mathematically
    ↪  sound, or if the step is a general statement or
    ↪  transition that does not contain reasoning. Respond
    ↪  with "incorrect" if the step includes any errors or
    ↪  flawed logic.

    ** Sotluion **
    We\'ll start from the beginning and proceed logically.'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'have "Jorge_current_age = 16
    ↪  + (2022 - 2005)" using assms by simp'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'then have "Jorge_current_age
    ↪  = 16 + 17" using assms by simp\nthen have
    ↪  "Jorge_current_age = 33" using assms by simp'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'then have "Simon_current_age
    ↪  = 33 + 24" using assms by simp'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'then have "Simon_current_age
    ↪  = 57" using assms by simp'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'then have "Simon_age_2010 =
    ↪  57 + (2010 - 2022 + 2005)" using assms by simp'},
    {'role': 'assistant', 'content': 'correct'},
    {'role': 'user', 'content': 'then have "Simon_age_2010 =
    ↪  57 + 83" using assms by simp'},
    {'role': 'assistant', 'content': 'incorrect'}
  ],
}
```

C Creation Process of FoVer-40K

This section provides details of the creation process of FoVer-40K, which is outlined in Section 3.



(a) Formal logic task.



(b) Formal theorem proving task.

Figure 6: Automatic step-level error annotation for FoVer-40K by formal verification tools.

C.1 Formal Logic

Formal solution generation (Appendix C.1.1). We prompt Llama 3.1 8B and Qwen 2.5 7B to generate step-by-step formal solutions to FLDx2 in a format compatible with Z3. We use a few-shot instruction to guide LLMs to follow the format and filter out syntactically invalid solutions using Z3.

Automatic error annotation (Appendix C.1.2). Z3 is designed for verification at the solution level, but we use it at the step level by supplying Z3 with the premises and conclusion for the target step to determine logical validity, as in Figure 6a.

C.1.1 Initial Response Generation

First, we need to generate symbolic solutions from LLMs, which may include logical mistakes, but they should be in a valid format compatible with Z3 (de Moura and Bjørner, 2008).

We use FLDx2 (Morishita et al., 2023, 2024) as a base dataset for our formal logic task. We use the symbolic version of the dataset. To simplify the verification pipeline, we removed cases whose reasoning steps include "assump," which is used in cases such as proof by contradiction.

We generate formal solutions from Llama 3.1 8B and Qwen 2.5 7B. The following is an example of a few-shot demonstration for the initial generation. We provide six examples as a demonstration. After generating the formal solutions, we filter out those with an invalid format using Z3.

```
{
  'role': 'user',
  'content':
    'Based on the provided facts ($context$), either prove or
    ↪  disprove the hypothesis or state that it is unknown.
    ↪  The facts and the hypothesis are written in logical
    ↪  formulas as follows: capital letters such as "{A}",
    ↪  "{B}", "{AB}" are predicates, small letters such as
    ↪  "{a}", "{b}", "{ab}" are constants, "&" is logical
    ↪  conjunction, "v" is logical disjunction, "¬" is
    ↪  negation, "->" is implication, "(x)" is "for all x",
    ↪  and "(Ex)" is "for some x".

    Don't generate anything other than the proof and
    ↪  proof_label.
    Don't generate the proof by contradiction. If the
    ↪  hypothesis is disproved, provide a direct
    ↪  counterexample. If the proof_label is PROVED, your
    ↪  proof should end with the 'hypothesis'. If the
    ↪  proof_label is DISPROVED, your proof should end with
    ↪  '¬hypothesis'.

    $hypothesis$: ({AB}{c} & ¬{B}{c})

    $context$:
    fact1: (x): ¬{D}x -> ({A}x & {C}x)
    fact2: ¬({B}{a} & ¬{A}{a})
    fact3: ¬{AB}{b} -> ¬{C}{d}
    fact4: ¬{C}{a} -> ¬{A}{d}
    fact5: ¬{A}{a}
    fact6: ({AB}{c} & ¬{B}{c}) -> {C}{d}
    fact7: ¬({F}{b} v {D}{b}) -> ¬{D}{a}
    fact8: {A}{a} -> ({AB}{c} & ¬{B}{c})
    fact9: ({AB}{c} & ¬{AA}{c}) -> {B}{b}
    fact10: ({AB}{c} & ¬{B}{c}) -> {C}{b}'
},
{
  'role': 'assistant',
  'content':
    '$proof$:
    fact6 -> int1: ¬{C}{d} -> ¬({AB}{c} & ¬{B}{c});
    fact3 & int1 -> int2: ¬{AB}{b} -> ¬({AB}{c} & ¬{B}{c});

    $proof_label$: UNKNOWN'
}
```

C.1.2 Automatic Step-level Annotation

Second, we automatically annotate step-level error labels, which will be used to train PRMs. We use Z3 (de Moura and Bjørner, 2008) for annotating step-level error labels. Specifically, we use the checker provided by the FLDx2 paper (Morishita et al., 2024) based on Z3.[1] Z3 is originally designed for solution-level verification, so we need to write code to use it for step-level verification. As explained in Figure 6a, we first postprocess each

[1] https://github.com/hitachi-nlp/FLD-generator/blob/00d12c4a9132a4fb43cd77f24db03ea7f5b27877/FLD_generator/formula_checkers/z3_logic_checkers/checkers.py#L179

step in solutions to an independent logical step and check the validity using Z3.

C.2 Formal Theorem Proving

Formal statement generation (Appendix C.2.1). We first generate formal statements in the format compatible with Isabelle from GSM8K-level problems, including GSM8K (Cobbe et al., 2021), GSM8K-based cases in MetaMathQA (Yu et al., 2024), and math word problems in Big-Math (Albalak et al., 2025), using Qwen 2.5 7B.

Formal solution generation (Appendix C.2.2). We instruct Qwen 2.5 7B to generate step-by-step formal proofs in the format compatible with Isabelle to the formal statements.

Automatic error annotation (Appendix C.2.3). Isabelle is designed for solution-level verification. To obtain step-level error labels, we implement wrapper code for step-level verification. Our code assumes that the other steps are correct when evaluating the target step.

C.2.1 Formal Statement Generation

First, we generate formal statements in the format compatible with Isabelle (Nipkow et al., 2002). Motivated by prior work (Zhou et al., 2024), we generate formal statements from math word problems using Qwen 2.5 7B with few-shot prompting. Specifically, we convert informal GSM8K-level math word problems: GSM8K (Cobbe et al., 2021), GSM8K-based cases in MetaMathQA (Yu et al., 2024), and math word problems in Big-Math (Albalak et al., 2025), into formal statements.

C.2.2 Formal Proof Generation

We provide an instruction with few-shot examples to generate formal proofs from Qwen 2.5 7B. The following is an example of a few-shot demonstration for the formal proof generation.

```
{
  'role': 'user',
  'content': '** Formal Statement **
theorem example:
    assumes "(Cabin_cost::nat) = 129000"
        and "(Gloria_cash::nat) = 150"
        and "(Cypress_count::nat) = 20"
        and "(Pine_count::nat) = 600"
        and "(Maple_count::nat) = 24"
        and "(Cypress_price::nat) = 100"
        and "(Maple_price::nat) = 300"
        and "(Pine_price::nat) = 200"
        and "(After_paying_Alfonso::nat) = Gloria_cash +
        ↪  Cypress_count * Cypress_price + Maple_count *
        ↪  Maple_price + Pine_count * Pine_price -
        ↪  Cabin_cost"
    shows "After_paying_Alfonso = 200"'
},
{
  'role': 'assistant',
  'content': 'proof -
    have "Cypress_count * Cypress_price = 2000"
        using assms by simp
    then have "Maple_count * Maple_price = 7200"
        using assms by simp
    then have "Pine_count * Pine_price = 120000"
        using assms by simp
    then have "Gloria_cash + Cypress_count * Cypress_price +
    ↪  Maple_count * Maple_price + Pine_count * Pine_price =
    ↪  129200"
        using assms by simp
    then have "Gloria_cash + Cypress_count * Cypress_price +
    ↪  Maple_count * Maple_price + Pine_count * Pine_price -
    ↪  Cabin_cost = 200"
        using assms by simp
    then have "After_paying_Alfonso = 200"
        using assms by simp
    show ?thesis
        using assms by simp
qed'
}
```

C.2.3 Automatic Step-level Proof Verification

Format verification. As we use LLMs for the conversion, it is possible that the conversion generates proofs in an invalid format. To filter out theorems in the invalid format, we use the "sorry" keyword of Isabelle. The "sorry" keyword serves as a placeholder for incomplete or unproven proofs, allowing the theorem to be accepted by the system without a formal justification. By inserting "sorry" into all generated proof steps, we can isolate and verify only the syntactic correctness theorems.

For example, in the following proof, `babysitting_minutes × (Weng_hourly_wage / 60)` contains the symbol `×`, which is not a valid multiplication operator in Isabelle syntax.

```
theorem example:
    assumes "(Weng_hourly_wage::real) = 12"
        and "(babysitting_minutes::real) = 50"
        and "(babysitting_hours::real) = babysitting_minutes
        ↪  / 60"
        and "(Weng_earnings::real) = Weng_hourly_wage *
        ↪  babysitting_hours"
    shows "Weng_earnings = 10"
proof -
    have "Weng_hourly_wage / 60 = 0.20"
        sorry
    then have "babysitting_minutes × (Weng_hourly_wage / 60)
    ↪  = 10"
        sorry
    then have "Weng_earnings = 10"
        sorry
    thus ?thesis
        sorry
qed
```

For this input, Isabelle returns the following error.

```
Step error: Inner syntax error (line 1)\nat \"? (
↪  Weng_hourly_wage / 60 ) = 10\"\nFailed to parse prop\nAt
↪  command \"have\" (line 1)
```

Step-level verification. By default, Isabelle halts at the first encountered error and does not provide a step-by-step verification of a proof. To enable independent verification of each step in a multi-step proof, we insert the "sorry" keyword in all but

one step. This allows Isabelle to type-check and parse each step individually, even if other steps are incomplete or invalid.

The following example is for verifying the third step independently. For each theorem, we run Isabelle once per step.

```
theorem example:
    assumes "(wallet_cost::nat) = 100"
        and "(betty_savings::nat) = wallet_cost div 2"
        and "(parent_contribution::nat) = 15"
        and "(grandparent_contribution::nat) = 2 *
        ↪  parent_contribution"
        and "(total_savings::nat) = betty_savings +
        ↪  parent_contribution + grandparent_contribution"
        and "(additional_needed::nat) = wallet_cost -
        ↪  total_savings"
    shows "additional_needed = 5"
proof -
    have "betty_savings = wallet_cost div 2"
        sorry
    then have "betty_savings = 50"
        sorry
    have "grandparent_contribution = 2 * parent_contribution"
        by simp
    then have "grandparent_contribution = 30"
        sorry
    then have "parent_contribution + grandparent_contribution
    ↪  = 45"
        sorry
    then have "total_savings = 95"
        sorry
    then have "additional_needed = wallet_cost -
    ↪  total_savings"
        sorry
    then have "additional_needed = 5"
        sorry
    thus ?thesis
        sorry
```

D FoVer Beyond FoVer-40K

As the first work towards this direction, we evaluate FoVer by creating PRM training data using relatively simple tasks with a minimal pipeline to keep the evaluation focused and clear. However, FoVer is not limited to the tasks and tools we used in this paper and can be extended to create training data using different types of tasks or more complex tasks. When applying FoVer to create PRM training data using more complex tasks, there are two potential challenges: (1) generating formal solutions from LLMs in a valid format compatible with formal verification tools and (2) verifying step-level correctness using formal verification tools.

First, formal solutions should be in a syntactically valid format compatible with formal verification tools we use for verification. Following syntactical rules and formats of some formal verification tools can be challenging for LLMs, especially when we target more complex problems. Recent LLMs are increasingly capable of generating formal solutions in valid formats, showing strong performance in first-order logic (Olausson et al., 2023) and a growing ability to produce syntactically valid formal proofs (Ren et al., 2025). We expect future models to further improve their capabilities to generate formal solutions and be more suitable for creating PRM training data using FoVer.

Second, we need to make formal verification tools verify step-level correctness. The tools are often designed for solution-level verification, so we often need to adapt them for step-level verification, as we did in this paper for creating FoVer-40K. When creating PRM training data using more complex tasks, we may need to further modify the verification pipeline to support new operations.

For example, to keep the verification pipeline simple, we did not use problems that involve assumptions in the formal logic task (FLDx2), such as proofs by contradiction, when creating FoVer-40K. However, we can extend our verification pipeline to support such cases. Existing verification tools are already capable of performing solution-level verification for proofs by contradiction, so we can make use of them to provide step-level verification. When handling assumptions in our framework, the type of mistake that cannot be detected through our current step-independent verification alone is illustrated by the following example, because the step-independent verification assumes that preceding intermediate results are correct:

- Premises: fact1: B; fact2: B→C; fact3: C→A;
- Hypothesis: A
- fact1 and fact2 → C; **Assume A; A → Hypothesis;** Therefore, the hypothesis is proved.

In this case, the existing solution-level verification will identify this solution as an error because the assumption is not properly discharged. Thus, by combining step-independent verification with solution-level verification, we can identify and label the final step as erroneous.

E Implementation Details

This section provides details of our experiments.

E.1 Input Format and Postprocessing

FoVer-PRMs and the baseline PRMs. We create inputs to LLM-based PRMs by preprocessing step-by-step solutions into a conversation format where each input contains a single step, and the expected output is a single token: "correct" or "incorrect". To obtain step-level scores, we extract logits for the two words and apply the softmax function to compute the prediction probability for

"correct". This approach follows prior work (Xiong et al., 2024) that uses LLMs as PRMs. As the baseline PRMs are not fine-tuned, we provide zero-shot instructions about this format.

First, we describe the input format for FoVer-PRMs and the baseline LLM-based PRMs, which are based on Llama 3.1 8B and Qwen 2.5 7B. FoVer-PRMs are trained on FoVer-40K, so the input format has the same format as the training data. The only difference is that we replace all step-level labels with "correct" in the input. This preprocessing allows us to provide the whole input once to get the step-level predictions for all steps. The following is an example input for GSM8K.

```
[
  {
    'role': 'user',
    'content': '** Problem **
      Alice is 7 years older than Beth, who is 5 years older
      ↪  than Erica. What is the difference between the ages
      ↪  of Alice and Erica, if Erica is 30 years old?

      ** Task **
      Your task is to evaluate the accuracy of each step in
      ↪  the provided solution to the above question. For
      ↪  each step, respond with "correct" if the reasoning
      ↪  is logically valid and mathematically sound, or if
      ↪  the step is a general statement or transition that
      ↪  does not contain reasoning. Respond with
      ↪  "incorrect" if the step includes any errors or
      ↪  flawed logic.

      ** Sotluion **
      Since Erica is 30 years old, and Beth is 5 years older
      ↪  than Erica, Beth is 30 + 5 = 35 years old.'
  },
  {'role': 'assistant', 'content': 'correct'},
  {'role': 'user', 'content': 'Alice is 7 years older than
  ↪  Beth, who is 35 years old. '},
  {'role': 'assistant', 'content': 'correct'},
  {'role': 'user', 'content': 'So, Alice is 35 + 7 = 42 years
  ↪  old.'},
  {'role': 'assistant', 'content': 'correct'},
  {'role': 'user', 'content': "The difference between Alice's
  ↪  age and Erica's age is 42 - 30 = 12 years."},
  {'role': 'assistant', 'content': 'correct'},
  {'role': 'user', 'content': 'Therefore, the answer (arabic
  ↪  numerals) is 12.'},
  {'role': 'assistant', 'content': 'correct'}
]
```

Next, we describe the postprocessing for FoVer-PRMs and the baseline LLM-based PRMs.

Extracting logits. Since we use causal LLMs as PRMs, we extract the model's predictions for the tokens immediately preceding the dummy step-level labels (e.g., "correct") in the input.

Computing step-level scores. At each identified position, we extract the logits corresponding to the tokens "correct" and "incorrect". We then apply the softmax function over these two logits to compute the probability assigned to "correct". This probability serves as the step-level score.

PRMs based on Llama 3.1 8B. In RLHFlow-Llama3.1-8B-DeepSeek and RLHFlow-Llama3.1-8B-Mistral (Xiong et al., 2024), the input format is mostly similar to ours, with the key difference being the use of "+" and "-" instead of "correct" and "incorrect".[2] For these models, we apply our input format and postprocessing procedures with a simple substitution of "correct" with "+".

PRMs based on Qwen 2.5 7B. Qwen2.5-Math-7B-PRM800K (Zheng et al., 2025) and Qwen2.5-Math-PRM-7B (Zhang et al., 2025) are supported by vLLM (Kwon et al., 2023). We follow the input format specified in their respective model descriptions and adopt the reward modeling in vLLM.[3] For Qwen2.5-7B-Skywork-PRM (He et al., 2024), we use a code provided by the authors.[4]

E.2 Training Settings

This section provides details of the training settings for our FoVer-PRMs.

Balanced labels. The last steps of the solutions in FoVer-40K includes balanced step-level error labels of 50% of "correct" and "incorrect" labels. Then, we can perform training on the balanced label by setting `mask_history: True` in LLaMA-Factory, which configures the model to use only the last step of each conversation during training.

Training parameters. We fine-tune all model parameters and do not use parameter-efficient techniques. We use the AdamW optimizer (Loshchilov and Hutter, 2019) and select the learning rate based on the average Best-of-K performance on the validation tasks: Orca-Math (Mitra et al., 2024) and two tasks in BBH (Suzgun et al., 2023) (Logical Deduction (three objects) and Boolean Expressions). We evaluate models trained with the learning rate 1e-6, 2e-6, 5e-6, and 1e-5, and select the model with the best average performance on the validation tasks. We use the parameters in Table 9 in all models, and we did not conduct hyperparameter tuning for these parameters. Please refer to the configuration files in our code for further details.

E.3 Evaluation Settings of Best-of-K

Benchmarks. Table 10 shows sources of reasoning benchmarks evaluated in Best-of-K in Section 4.

Initial generation prompts. Table 11 shows de-

[2] https://github.com/RLHFlow/RLHF-Reward-Modeling/tree/main/math-rm

[3] https://docs.vllm.ai/en/latest/models/pooling_models.html

[4] https://github.com/SkyworkAI/skywork-o1-prm-inference

Parameter	Value
Number of Epochs	1
Batch size	32
Learning Rate Warm up and Decay Strategy	Linear
Learning Rate Warm up Ratio	0.5

Table 9: Hyperparameters for training on FoVer-40K

tailed settings of generating $K = 7$ responses for the Best-of-K evaluation in Section 4. We create new few-shot examples or modify few-shot demonstrations used in prior work to enhance the quality and to simplify the post-processing procedure. For example, we add line breaks between reasoning steps in all tasks. An example prompt for GSM8K is provided in Appendix C.1.1. Please also refer to our code for further details.

E.4 Computational Resources

We use four NVIDIA A100 SXM4 80GB GPUs for training and inference. Training each 8B-class model on our dataset takes approximately one hour using our training data, FoVer-40K. Evaluation takes considerably more time because the Best-of-K evaluation requires multiple candidate solutions, and reproducing all the evaluation in this paper will take approximately three days.

Verification using Z3 and Isabelle does not require GPUs and can be run in parallel on CPU-only servers, enabling efficient and scalable PRM training data creation. On average, in a single process, Z3 verifies all datasets in less than 10 minutes for the formal logic task, and Isabelle verifies each step in approximately 3 seconds for the formal theorem proving tasks. For annotation, we run one Z3 process for formal logic and 40 parallel Isabelle processes for formal theorem proving on AMD EPYC 7763 64-core processors.

E.5 Model Access and Software Libraries

LLMs. We use LLMs from Hugging Face Hub. We use `meta-llama/Llama-3.1-8B-Instruct` and `Qwen/Qwen2.5-7B-Instruct` as base models for our PRMs. We also use these models to generate initial responses used in creating FoVer-40K, and also for generating $K = 7$ responses in the Best-of-K evaluation (§4).

Existing PRMs. Details for the PRMs evaluated in Section 4 are listed in Table 12. We acquire these models at Hugging Face Hub and use vLLM (Kwon et al., 2023) to generate reward scores.[7]

Software libraries. For inference, we use vLLM (Kwon et al., 2023) for accelerating LLM inference. For training, we use LLaMA-Factory (Zheng et al., 2024) for fine-tuning PRMs.

F Ethical Considerations

Social impacts. This research does not involve human subjects or private data. This paper proposes a method for improving PRMs, which detect mistakes in LLM responses. We do not expect any harmful impact.

Dataset. Our dataset, FoVer-40K, is created from formal verification tools and logic and mathematical reasoning datasets. We do not expect that our dataset includes information that uniquely identifies individual people or offensive content.

The use of Large Language Models. In paper writing, we used LLMs to polish writing. We used ChatGPT-5 via OpenAI's web interface.

G Reproducibility Statement and License

The datasets, models, and code are provided at https://github.com/psunlpgroup/FoVer.

We release our dataset under Creative Commons Attribution 4.0 International and our code under Apache License 2.0. Our dataset and code are based on the following resources. We consider our license to be (one-way) compatible with all licenses listed below.

Datasets. FoVer-40K is based on the following datasets.

- FLDx2 (2024): CC BY 4.0[8]
- GSM8K (2021): MIT[9]
- MetaMathQA (2024): MIT[10]
- Big-Math (2025): Apache License 2.0[11]

Code and packages. Our code is partially based on the following resources.

[7] https://docs.vllm.ai/en/latest/models/supported_models.html#reward-modeling-task-reward

[8] https://github.com/hitachi-nlp/FLD-corpus/blob/neurips_2025/LICENSE

[9] https://github.com/openai/grade-school-math/blob/master/LICENSE

[10] https://huggingface.co/datasets/meta-math/MetaMathQA/blob/main/README.md

[11] https://huggingface.co/datasets/SynthLabsAI/Big-Math-RL-Verified/blob/main/README.md

Dataset	Source
GSM8K	https://huggingface.co/datasets/openai/gsm8k
MATH	https://github.com/hendrycks/math
AQuA-RAT	https://huggingface.co/datasets/deepmind/aqua_rat
AIME (2016-2024)	https://huggingface.co/datasets/di-zhang-fdu/AIME_1983_2024
FOLIO	https://huggingface.co/datasets/yale-nlp/FOLIO
LogicNLI	https://huggingface.co/datasets/tasksource/LogicNLI
ANLI	https://huggingface.co/datasets/facebook/anli
HANS	https://github.com/tommccoy1/hans
MMLU	https://huggingface.co/datasets/sam-paech/mmlu-pro-nomath-sml
BBH	https://github.com/suzgunmirac/BIG-Bench-Hard/tree/main/bbh

Table 10: Reasoning benchmarks evaluated in the Best-of-K experiments.

Dataset	Few-shot Examples for Initial Generation	Answer Matching
GSM8K	Kojima et al. (2022) [5]	Exact match after extraction and conversion to integer
MATH	Lewkowycz et al. (2022, Appendix D.2)	Lewkowycz et al. (2022, Appendix G)
AQuA-RAT	Made by us (3-shot)	Exact match after extraction
AIME	Made by us (3-shot)	Exact match after extraction and conversion to integer
FOLIO	Made by us (2-shot)	Exact match after extraction
LogicNLI	Made by us (3-shot)	Exact match after extraction
ANLI	Made by us (3-shot)	Exact match after extraction
HANS	Made by us (2-shot)	Exact match after extraction
MMLU	Made by us (4-shot)	Exact match after extraction
BBH	Suzgun et al. (2023)[6]	Exact match after extraction

Table 11: Detailed settings for Best-of-K downstream evaluation

PRMs	Source	Base Datasets	Training Data Creation
RLHFlow-Llama3.1-8B-DeepSeek (2024)	RLHFlow/Llama3.1-8B-PRM-Deepseek-Data	GSM8K, MATH	Math-Shepherd (2024)
RLHFlow-Llama3.1-8B-Mistral (2024)	RLHFlow/Llama3.1-8B-PRM-Mistral-Data	GSM8K, MATH	Math-Shepherd (2024)
Qwen2.5-Math-7B-PRM800K (2025)	Qwen/Qwen2.5-Math-7B-PRM800K	MATH	Human annotation
Qwen2.5-Math-PRM-7B (2025)	Qwen/Qwen2.5-Math-PRM-7B	Private Data	Math-Shepherd (2024) & LLM-as-a-Judge
Qwen2.5-7B-Skywork-PRM (2024)	Skywork/Skywork-o1-Open-PRM-Qwen-2.5-7B	Hidden	Hidden

Table 12: PRMs we evaluate in Section 4.

- FLD (2024): Apache License 2.0[12]
- Isabelle: BSD-style regulations[13]
- Neural theorem proving tutorial (2023): MIT[14]
- DTV (2024): MIT[15]

[12] https://github.com/hitachi-nlp/FLD/blob/neurips_2025/LICENSE

[13] https://isabelle.in.tum.de/

[14] https://github.com/wellecks/ntptutorial/blob/main/LICENSE

[15] https://github.com/jinpz/dtv/blob/main/LICENSE