Process Reward Models Meet Planning: Generating Precise and Scalable Datasets for Step-Level Rewards

Raffaele Pisano
Babelscape
pisano@babelscape.com

Roberto Navigli
Babelscape & Sapienza University of Rome
navigli@diag.uniroma1.it

Abstract

Process Reward Models (PRMs) have emerged as a powerful tool for providing step-level feedback when evaluating the reasoning of Large Language Models (LLMs), which frequently produce chains of thought (CoTs) containing errors even when the final answer is correct. However, existing PRM datasets remain expensive to construct, prone to annotation errors, and predominantly limited to the mathematical domain. This work introduces a novel and scalable approach to PRM dataset generation based on planning logical problems expressed in the Planning Domain Definition Language (PDDL). Using this method, we generate a corpus of approximately one million reasoning steps across various PDDL domains and use it to train PRMs. Experimental results show that augmenting widely-used PRM training datasets with PDDL-derived data yields substantial improvements in both mathematical and non-mathematical reasoning, as demonstrated across multiple benchmarks. These findings indicate that planning problems constitute a scalable and effective resource for generating robust, precise, and fine-grained training data for PRMs, going beyond the classical mathematical sources that dominate this field.

1 Introduction

Recently, LLMs have achieved remarkable progress across a wide range of domains, including mathematics, logic, and program synthesis (OpenAI, 2024; Guo et al., 2025; Qwen Team, 2025b; Mistral AI, 2025; Google DeepMind, 2025). Models that use CoT reasoning have shown clear advantages over those that do not; however, they still frequently exhibit inconsistencies in their intermediate reasoning steps (Turpin et al., 2023; Lightman et al., 2024; Zheng et al., 2025; Stechly et al., 2025a). It is not uncommon for a model to produce a correct final answer while the accompanying reasoning contains flawed, illogical, or self-contradictory steps. These observations highlight the importance of evaluating reasoning at the process level, rather than relying solely on final-answer correctness, in order to better detect and mitigate logical imperfections and improve the robustness of LLM reasoning.

Process Reward Models (PRMs) have gained attention as a promising approach to addressing this challenge (Uesato et al., 2022). By assigning step-level rewards, PRMs enable a detailed evaluation of individual steps within CoTs. However, existing PRMs are trained on datasets that present some limitations. One of the most notable datasets is PRM800k (Lightman et al., 2024), created through manual annotation of reasoning steps, a process that is time-consuming and difficult to scale. The dataset also provides limited granularity, distinguishing steps only as good, neutral, or bad, and focuses exclusively on mathematical reasoning. Annotators were required to satisfy a minimum agreement threshold, leaving room for imperfect or unreliable labels.

To avoid human annotation effort, Wang et al. (2024b) introduce the Math-Shepherd dataset, constructed via an LLM-based reward annotation approach. However, this method is computationally expensive and yields only a coarse estimate of step-level correctness. In particular, annotations are restricted to binary labels (good vs. bad) and are confined to mathematical problem domains, without extending to broader forms of logical reasoning. These limitations reduce data quality and limit potential improvements for PRMs.

To fill these gaps, we propose a novel framework for PRM dataset generation based on the Planning Domain Definition Language (PDDL), a formal language for representing a subset of reasoning problems, namely planning problems. Our approach enables automatic, rule-based reward assignment and the creation of large-scale, high-precision reasoning datasets that extend beyond mathematical tasks.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28022–28042

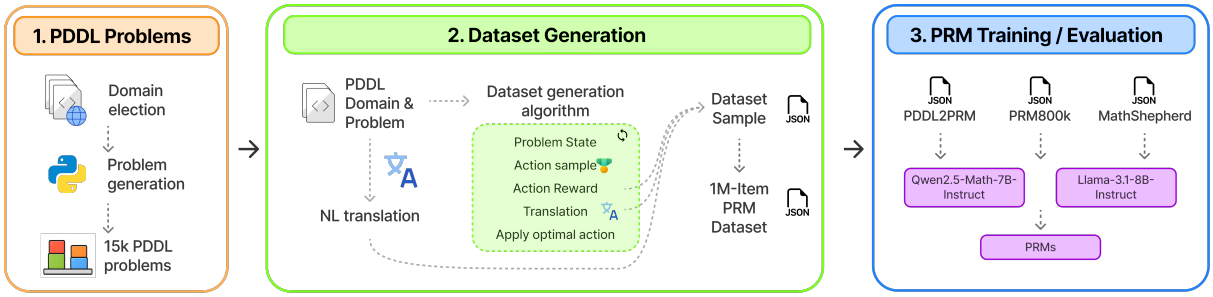

Figure 1: End-to-end workflow of the proposed framework: from PDDL problem generation, to dataset construction with step-level rewards, to PRM training.

Empirical results show that integrating PDDL-derived data with PRM800k and Math-Shepherd substantially improves PRMs' performance, especially on non-mathematical benchmarks. Our key contributions are:

- Introduction of a novel pipeline for PRM dataset generation based on PDDL, providing a scalable and reproducible framework for reasoning supervision.
- Generation of a large-scale dataset of around one million reasoning steps across eleven PDDL domains, covering approximately 15,000 planning problems.
- Training, evaluation, and release of multiple PRMs with competitive performance, enabling further experimentation and evaluation.
- Comparison between regression- and classification-based PRM training, showing the advantages of the regression formulation.
- Demonstration that PDDL-augmented training improves both accuracy and robustness in step-level reasoning evaluation, extending generalization beyond mathematical domains.

Overall, this work establishes PDDL-based dataset generation as a promising direction for developing more structured, interpretable, and scalable reasoning supervision. The datasets and the trained PRMs used in this work are available at the following link: `https://github.com/Babelscape/prm-meets-planning/`. The whole workflow is shown in Figure 1.

2 Related Work

2.1 Feedback in Reasoning LLMs

The notable progress of LLMs in reasoning tasks (OpenAI, 2024; Guo et al., 2025; Qwen Team, 2025b; Mistral AI, 2025; Google DeepMind, 2025), together with the emergence of Long Chain of Thought (Chen et al., 2025), has attracted significant attention to their cognitive and problem-solving capabilities. Within this context, feedback plays a crucial role, serving as an external signal that can be used to assess and refine CoT-based reasoning.

An important distinction has been drawn between process feedback, which performs step-level evaluation through PRMs, and outcome feedback, which assesses only the final result through Outcome Reward Models (Uesato et al., 2022, ORMs). PRMs have been shown to be more effective than ORMs in enhancing reasoning performance, particularly in mathematical tasks (Lightman et al., 2024; Luo et al., 2024; Sun et al., 2024). However, outcome-level evaluation is comparatively straightforward to implement, whereas process-level evaluation is inherently more complex.

2.2 Process Reward Models

PRMs have been introduced to enable process-level evaluation, focusing on assessing the intermediate reasoning steps rather than only the final answers. However, their effectiveness is constrained by the quality and diversity of the datasets currently available for their training, the creation of which remains a challenging open problem. Several approaches have been proposed for generating datasets to support PRM training. Lightman et al. (2024) introduced a large-scale dataset in the mathematical domain, where reasoning steps are manually annotated as good, neutral, or bad. While widely adopted, this approach suffers from limited scalability due to the high cost of manual annotation. Moreover, the annotators selected for the final dataset were chosen based on achieving over 75% agreement with gold labels, implying that up to 25% of the final annotations may still contain

errors. Consequently, while the dataset represents a valuable resource, its reliability is not guaranteed.

The Math-Shepherd dataset (Wang et al., 2024b), on the other hand, avoids the need for human annotators by adopting an automatic reward assignment method. For each intermediate step in a reasoning chain, the solution is truncated at that point and multiple continuations are generated using an LLM. The quality of a step is then estimated as the fraction of these continuations that reach the correct final answer. Although this strategy is effective, it remains both computationally expensive and indirect. As a result, the dataset lacks gold-standard supervision, as it evaluates utility rather than intrinsic correctness, and is ultimately restricted to a binary classification (correct vs. incorrect).

Other approaches, such as PRIME (Cui et al., 2025) and FPRWL (Yuan et al., 2025), bypass explicit annotations by deriving implicit signals from the correctness of final solutions, for example using log-likelihood scores. Efficiency improvements have also been investigated. EPIC-PRM (Sun et al., 2025) reduces computational costs while maintaining annotation quality. Similarly, Zhang et al. (2025) combine the Math-Shepherd strategy with an additional LLM acting as a "judge" to double check step quality. More recently, Molfese et al. (2026) show that state-of-the-art PRMs struggle to generalize beyond mathematics, particularly to commonsense reasoning tasks. Finally, VersaPRM (Zeng et al., 2025) extends the scope of PRM dataset generation to domains such as physics, law, and economics. However, it still draws on an annotation procedure that employs an LLM as a step-level judge, making it potentially error-prone, and uses only three labels (good, neutral, bad).

Our work similarly extends the scope of PRM dataset generation beyond the mathematical domain, leveraging logical planning problems across diverse types of task. However, unlike prior approaches, our method assigns rewards using precise, rule-based criteria and provides a richer set of reward levels, thereby addressing key limitations of existing techniques in terms of scalability, precision, and label granularity.

2.3 PRMs and PDDL

PDDL is a formal language designed to describe planning domains and problems in a structured and machine-interpretable format. It provides a standardized framework for representing states, actions, and goals, making it a fundamental tool in automated planning and reasoning research. An increasing number of works have explored the intersection between PDDL and LLMs. Among these, several contributions highlight the potential of planning problems as a testbed for evaluating the reasoning abilities of LLMs. In particular, the works by Valmeekam et al. (2023a,b); Vyas et al. (2025); Goebel and Zips (2025) examine the capabilities of LLMs in planning and action reasoning, including their ability to generate and validate simple plans. Stechly et al. (2024, 2025b) show that LLMs struggle substantially with self-correction when engaged in planning-related tasks, and that the use of CoT offers only limited improvements in their ability to reason through planning problems. In a different direction, Verma et al. (2025) propose a framework in which LLMs are trained not only to output a plan but also to rigorously reason about action applicability, state transitions, and plan validity. Other approaches (Guan et al., 2023; Oswald et al., 2024) do not use PDDL to evaluate LLMs; instead, they utilize them to generate PDDL problems that can then be solved by external symbolic planners. In contrast to these studies, our work neither relies on LLMs to generate PDDL problems nor uses PDDL directly to train or evaluate them.

3 PDDL for LLM Reasoning Evaluation

Our core idea is to exploit the nature of PDDL not only as a source of logical reasoning problems for evaluating reasoning capabilities of LLMs, as previously done in the literature, but also as a structured source for constructing artificial reasoning chains with step-level rewards, which are subsequently used to train PRMs. The final goal is to create a scalable and low-cost dataset in which each sample includes a PDDL problem expressed in natural-language form, together with a partially solved reasoning trajectory. The trajectory consists of a variable number of intermediate steps, each aligned with a specific action and associated with a precise reward. The large number of domains and problem instances that can be generated in PDDL naturally enables scalable dataset construction.

Formalization of Planning Concepts To clarify the theoretical background underlying our work, we first introduce the fundamental concepts of Classical Planning (Ghallab et al., 2004).

A *planning domain* is defined as a tuple:

$$D = \langle S, A, \delta \rangle$$

where S is a finite set of states, A is a finite set of actions, and $\delta : S \times A \to S$ is a deterministic transition function that specifies the successor state resulting from applying an action in a given state.

A *planning problem* is described by the triple:

$$P = (D, s_0, G),$$

where $s_0 \in S$ is the initial state and $G \subseteq S$ denotes the set of goal states to be achieved.

A *plan* is a finite sequence of actions:

$$\pi = \langle a_1, a_2, \ldots, a_n \rangle$$

such that each transition satisfies $s_{i+1} = \delta(s_i, a_{i+1})$ for $i = 0, \ldots, n-1$, and the final state $s_n \in G$.

A *planner* is an algorithm that, given a problem P, computes a plan π leading from s_0 to a state that satisfies the goal condition.

Dataset Generation Framework At the core of our approach lies the interpretation of a planning problem as a logical reasoning task, where solving the problem corresponds to finding a sequence of actions that leads from an initial state to a goal state. From this perspective, the plan represents the reasoning process itself, and each action corresponds to an individual reasoning step. To generate the dataset we therefore simulate multiple planning steps and assign a reward to each action according to specific evaluation rules. The following utility functions are introduced to facilitate dataset construction:

- `get_rand_action(P,s)` $\to a$, which returns a random action $a \in A$, drawn from a distribution that includes both applicable and non-applicable actions in state $s \in S$.
- `get_opt_action(P,s)` $\to a$, which returns the first action $a \in A$ of the optimal plan starting at state s. The optimal plan is computed by an external planner that guarantees optimality (see implementation details in Section 4.2.1).
- `eval_action(P,s,a)` $\to r$, which returns a reward $r \in [0, 1]$ evaluating the quality of action a in state s, based on its executability, proximity to the goal, and whether it belongs to the optimal plan. Further implementation details are provided in Section 4.2.2.
- `translate(P,s,a,r)`, which converts the action into a natural language reasoning step and adds it to the dataset with reward r.

Algorithm 1 Dataset Generation

Require: $P = ((S, A, \delta), s_0, G)$
$s \leftarrow s_0$
while $s \notin G$ **do**
  for $i = 1$ to y **do**
    $a \leftarrow$ `get_rand_action`(P, s)
    $r \leftarrow$ `eval_action`(P, s, a)
    `translate`(P, s, a, r)
  end for
  $a^* \leftarrow$ `get_opt_action`(P, s)
  $s \leftarrow \delta(s, a^*)$
end while

Algorithm 1 illustrates the dataset generation process. At each iteration, y random actions are sampled, where y is a domain-dependent parameter controlling how many exploratory steps are generated per state. The sampled actions are then evaluated using `eval_action(P,s,a)` and translated using `translate(P,s,a,r)` for inclusion in the dataset. The procedure is repeated until the goal state is reached (i.e. $s \in G$). By sampling diverse candidate actions, including suboptimal, poor, and optimal ones, the algorithm ensures that each state is associated with a rich set of alternative reasoning steps, creating both correct and incorrect trajectories.

4 Experimental Setup

4.1 Planning Domain Overview

We use 11 different PDDL domains, most of which originate from the International Planning Competition (Fox and Long, 2002, IPC), while others are custom-designed by drawing inspiration from existing domains. The domains span a diverse set of planning tasks, which we group into the following categories.

Manipulation and Rearrangement Tasks. This category includes domains where an agent rearranges stacked objects to reach a target configuration (*BlocksWorld-3-ops*, *BlocksWorld-4-ops*).

Transportation Tasks. These domains involve moving entities across locations optimizing transportation efficiency (*Ferry*, *Logistics*, *Elevator*).

Puzzles and Constraint Satisfaction Tasks. This group consists of classical puzzle problems that require planning under strict movement constraints (*Tower of Hanoi*, *N-Puzzle*).

Category	Description	Reward
Non-executable	a cannot be applied because its preconditions are not satisfied in S	0.0
Dead-end	a leads to a state S' from which the goal is unreachable	0.25
Backtracking	a leads to a state S' from which the optimal plan requires revisiting a previous state	0.5
Suboptimal	a is valid but does not belong to any optimal plan	0.75
Optimal	a belongs to at least one optimal plan starting from S	1.0

Table 1: Category definitions and reward assignment. Each category is associated with a reward value reflecting the correctness of an action a in a given starting state S and its contribution to the reasoning process.

Navigation and Exploration Tasks. Domains in this category focus on spatial reasoning and sequential decision-making, often involving irreversible actions (*VisitGrid*, *Sokoban*, *Rooms*, *Spanner*).

Further details are provided in Appendix A.

4.2 Dataset Generation details

4.2.1 PDDL Processing

For handling and manipulating PDDL problems, we rely on the UnifiedPlanning library (Micheli et al., 2025) which provides a framework to parse PDDL domain and problem files, query the set of applicable actions in each state, and interface with external planners. The `get_opt_action(P,s)` function, previously introduced, computes the optimal action through the use of the Fast Downward planner (Helmert, 2006) configured with A* search and the LM-Cut heuristic (Helmert and Domshlak, 2009). LM-Cut is admissible, meaning that it never overestimates the true cost-to-go to the goal, and therefore guarantees optimal solutions under A* search (Dechter and Pearl, 1985), ensuring that the first action extracted from the plan is optimal.

4.2.2 Action Reward Assignment

To the best of our knowledge, there is no single, universally accepted taxonomy of action types in planning. However, several distinctions are useful for the purposes of our work. Beyond the basic separation between executable, non-executable, optimal and sub-optimal actions, we identify two additional categories that are particularly relevant. First, we consider actions that lead to states from which the goal is no longer reachable (Atkins et al., 1997, dead-end states). Second, we consider actions that induce backtracking, namely actions that lead to states from which an optimal plan requires revisiting a previously visited state. Based on these considerations, we adopt a taxonomy of five action categories, each associated with a scalar reward in the range $[0, 1]$ reflecting the degree of progress an action provides toward the goal state, as summarized in Table 1.

Accordingly, the previously introduced `eval_action(P,s,a)` function is implemented to determine which of the five categories the action `a` belongs to when executed in state `s`. Using the planner and the PDDL processing tools, the function first checks whether the action is *Non-executable* in the current state. If the action is executable, the successor state is computed and assessed to determine whether the planner can still find an optimal plan from it; if this is not possible, the action is labeled as a *Dead-end*. If an optimal plan does exist from the successor state, the function checks whether this plan revisits a previously encountered state, which characterizes *Backtracking*. Finally, if action `a` belongs to at least one optimal plan, it is labeled as *Optimal*; otherwise, it is classified as *Suboptimal*.

4.2.3 Dataset Composition and Splits

To evaluate the models in a completely unseen scenario and assess their generalization capabilities, the *Rooms* environment is used as a held-out test domain and therefore excluded from the training set. The remaining data are split into training, validation, and test sets according to an 85%–5%–10% ratio. The resulting dataset, which we name PDDL2PRM, contains approximately one million reasoning steps, each annotated with a reward, providing a comprehensive resource for training and evaluating PRMs.

4.2.4 Statistics

Because each planning domain exhibits intrinsic structural differences, both the number of generated problems and the length of their optimal solutions vary substantially. In many domains, increasing the number of objects, relations, or spatial degrees of freedom would in principle allow the generation of additional problems, but it would at the same time significantly increase the complexity

Domain	Problems	MOPL	Total steps
BlocksWorld-3	952	5.29	51,990
BlocksWorld-4	1,796	9.58	125,402
Ferry	858	9.12	65,269
Hanoi	1,660	3.84	65,745
Logistics	669	8.70	70,238
Elevator	2,089	10.53	207,549
N-Puzzle	598	9.05	42,027
Rooms	916	6.59	37,947
Sokoban	437	14.25	58,475
Spanner	3,563	7.87	198,942
VisitGrid	1,176	5.86	61,390
Total	**14,714**	**7.94**	**984,974**

Table 2: Statistics for each domain, including the number of problems, the Mean Optimal Plan Length (MOPL), and the total number of alternatives (steps).

of the resulting instances, often leading to much longer optimal plans. As a consequence, scaling the number of generated problems is not always practical, since we aim to avoid excessively long plans, which would produce reasoning trajectories that are difficult for models to handle. This effect is particularly evident in domains such as *Sokoban*, where the agent must navigate to precise grid locations before pushing boxes toward their targets. As the number of boxes increases, the sequence of required navigations and pushes grows accordingly, causing the optimal plan length to increase rapidly. Conversely, we observe that domains such as *Spanner* and *BlocksWorld* allow a broader variety of configurations without inducing the same degree of growth in plan length. The resulting statistics for each domain are reported in Table 2.

4.2.5 Computational Cost and Scalability of Optimal Planning

From the perspective of computational cost and scalability, our approach offers several advantages.

First, dataset generation does not require GPU utilization for large-scale sampling from LLMs, which is often the dominant cost in alternative PRM supervision approaches. Instead, it is entirely CPU-bound, performed offline, and fully parallelizable across problem instances and domains.

Second, in the regime relevant for PRM training, optimal planning does not constitute a significant computational bottleneck. PDDL enables controlled domain design and systematic instance generation across varying difficulty levels, allowing us to regulate plan length and avoid instances that would be excessively costly for the planner.

We intentionally exclude domains and instances with extremely long optimal plans. This decision is not primarily driven by planning cost, but by reasoning considerations: long plans and large domains yield trajectories that are disproportionately difficult for PRMs to learn from. In practice, LLM reasoning degrades significantly as plan length increases (Goebel and Zips, 2025; Stechly et al., 2024).

More broadly, our goal is not to solve arbitrarily complex planning problems, but to generate large quantities of structured and logically coherent reasoning trajectories. This is achieved by sampling diverse yet tractable domains and instances.

Consequently, within the range of problem sizes relevant for PRM training, the effective complexity limit is dictated more by the reasoning capacity of PRMs than by the computational limits of the planner.

4.3 Training Setup

To increase the robustness and reliability of our results, we conduct experiments training two different models: *Qwen2.5-Math-7B-Instruct* (Yang et al., 2024; Qwen Team, 2025a) and *Llama-3.1-8B-Instruct* (Llama Team, 2024; Meta AI, 2024), whose sizes align with the scale of most PRMs reported in the literature (Chen et al., 2025).

To evaluate the effectiveness of PDDL2PRM, we first train both models on a commonly adopted dataset (PRM800k or Math-Shepherd) as a point of comparison, and then retrain them on an augmented corpus that combines the same dataset with our PDDL-derived data. Importantly, since PRM800k includes mathematical problems derived from the MATH test set, and one of our evaluation benchmarks, ProcessBench (Zheng et al., 2025), is based on the same test samples, we follow the filtering approach adopted in ProcessBench and remove all MATH test samples from PRM800k to prevent train–test contamination.

Our training strategy follows a head-based architecture inspired by Zhang et al. (2025), which we adopt due to the strong performance reported in their work. In this setup, a head is added to the decoder to predict a scalar reward for each reasoning step. Unlike their approach, which formulates the task as classification, our PDDL-derived dataset provides multiple graded reward levels, enabling training PRMs as regressors. For comparison, we also train models using a classification loss to assess the effectiveness of the two formulations. Each trained model is identified by a naming convention

that specifies the underlying model, the training dataset(s), and a suffix indicating the learning objective, with `-r` denoting regression and `-c` denoting classification. Further implementation details are provided in Appendix B.

We choose not to train the models exclusively on PDDL2PRM. Although the automatically translated actions are semantically accurate, they follow translation templates and lack the fluency and naturalness of CoT reasoning. For this reason, we find PDDL-derived data most effective when combined with existing resources, as it provides precise reasoning signals while other datasets contribute more natural and expressive reasoning patterns. Empirical results for a model trained solely on PDDL-derived data are reported in Appendix C.

4.4 Test Data

For evaluation, we use three different benchmarks.

ProcessBench (Zheng et al., 2025) measures the ability to identify erroneous steps in mathematical CoT reasoning; given a math problem and a step-by-step solution, models must either identify the earliest-occurring error or conclude that all steps are correct. To evaluate model performance, the benchmark reports accuracy separately for instances containing at least one error in the CoT and for those with a fully correct reasoning chain. It then computes their harmonic mean as an F1 score, providing a balanced measure of the model's sensitivity to reasoning errors versus its tendency to incorrectly flag correct reasoning.

The second benchmark, **PRMBench** (Song et al., 2025), is designed to evaluate PRMs by comparing their rewards against gold-standard annotations, providing a comprehensive assessment through metrics such as simplicity, soundness, and sensitivity. The final score differs slightly from the previous benchmark, as it is computed as the arithmetic mean of the F1 score on instances containing erroneous reasoning steps and the F1 score on instances with fully correct CoT. Since PRMBench is constructed from PRM800k, all models trained on it are excluded from this evaluation.

MR-Ben (Zeng et al., 2024) evaluates PRMs across multiple domains, including physics, science, logic, and medicine. In its full setting, the benchmark requires models not only to identify the earliest erroneous step in a CoT, but also to provide an explanation of the error and a corresponding correction. However, given the structural constraints of the PRMs considered in this study, which output only scalar rewards, these explanatory and corrective components cannot be evaluated. Consequently, we restrict the evaluation to the error-identification task and score models using the same F1 metric as that described for ProcessBench.

Finally, we perform an additional evaluation on the PDDL2PRM test set and on previously excluded samples from the *Rooms* domain to assess the ability of the PRMs to judge reasoning chains derived from PDDL problems. For consistency with the previous benchmarks, the models are tasked with identifying the first erroneous step in the CoT. Performance is measured using the same F1 metric as in ProcessBench and MR-Ben.

5 Results

We compare against the following models, all fine-tuned using *Qwen2.5-Math-7B-Instruct* and selected as the most competitive existing PRMs with comparable parameter sizes:

- **Skywork-PRM-7B** (He et al., 2025): with no details disclosed about the training procedure.
- **Qwen2.5-Math-7B-PRM800k** (Zheng et al., 2025): trained as a binary classifier on the PRM800k dataset, where neutral annotations are mapped to the positive class. As anticipated, we fine-tune the same model on the same dataset while preserving the original labels and adopting a regression loss; our fine-tuned model is distinguished by the `-r` suffix.
- **Qwen2.5-Math-PRM-7B** (Zhang et al., 2025): trained on a large synthetic, non-public mathematical dataset and currently the strongest-performing PRM at this model scale. For experimental purposes, we further fine-tune this model on our PDDL-derived data.

Mathematical Reasoning Performance. On ProcessBench, as reported in Table 3, PRMs fine-tuned on the combined dataset that includes PDDL-derived data achieve an average improvement of +13.4% compared to their non-PDDL counterparts. The gains are particularly pronounced for models trained on Math-Shepherd, where the improvement is consistently larger. Similar trends are observed on PRMBench, as shown in Table 4, with an average performance increase of +8.0%. These results are particularly relevant as they indicate that training PRMs to evaluate planning-oriented reasoning

Model	GSM8K			MATH			OlympiadBench			Omni-MATH			Avg. F1
	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	
Skywork-PRM-7B	61.8	82.9	70.8	43.8	62.2	53.6	17.9	31.9	22.9	14.0	41.9	21.0	42.1
Qwen2.5-Math-PRM-7B	72.0	**96.4**	82.4	68.0	**90.4**	**77.6**	55.7	85.5	67.5	55.2	83.0	66.3	73.5
†Qwen2.5-Math-PRM-7B-PDDL-r	**73.4**	95.9	**83.2**	**68.2**	89.2	77.3	**57.9**	81.1	**67.6**	**57.6**	78.8	**66.5**	**73.6**
Qwen2.5-Math-7B-PRM800k	53.1	95.3	68.2	48.0	90.1	62.6	35.7	**87.3**	50.7	29.8	**86.1**	44.3	56.5
†Qwen2.5-Math-7B-PRM800k-r	58.0	89.6	70.4	60.8	74.4	66.9	48.4	56.0	52.0	49.7	57.3	53.2	60.6
†Qwen2.5-Math-7B-PRM800k-PDDL-r	67.6	90.2	77.3	67.2	73.4	70.1	51.7	53.1	52.4	50.6	52.7	51.6	62.9
†Llama-3.1-8B-PRM800k-r	45.9	91.2	61.1	46.0	58.1	51.3	43.7	31.3	36.5	42.6	35.3	38.6	46.9
†Llama-3.1-8B-PRM800k-PDDL-r	58.5	91.7	71.4	53.9	55.4	54.6	46.3	27.1	34.2	42.7	32.4	36.8	49.3
†Qwen2.5-Math-7B-MathShepherd-r	51.7	96.9	67.4	20.0	96.8	33.2	7.3	95.9	13.5	4.0	96.7	7.6	30.4
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	58.0	93.8	71.7	34.7	88.9	49.9	19.2	81.1	31.1	13.6	79.7	23.2	44.0

Table 3: Results on ProcessBench. † indicates PRMs we trained; best results in **bold**.

Model	Simplicity	Soundness	Sensitivity	Positive F1	Negative F1	Avg. F1
Skywork-PRM-7B	**81.7**	56.6	**90.1**	89.2	40.9	65.1
Qwen2.5-Math-PRM-7B	52.1	71.0	75.5	**91.5**	39.4	65.5
†Qwen2.5-Math-PRM-7B-PDDL-r	55.3	**72.3**	60.2	90.9	**44.2**	**67.5**
†Qwen2.5-Math-7B-MathShepherd-r	48.5	55.1	51.9	81.5	26.5	54.0
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	52.3	63.8	56.5	86.8	35.3	61.0

Table 4: Results on PRMBench. † indicates PRMs we trained; best results in **bold**. The full results on this benchmark are reported in Appendix E.

confers transferable benefits to mathematical reasoning evaluation, despite the PDDL-based data not being intrinsically mathematical.

Generalization Beyond Mathematics. Results on MR-Ben (Table 5) show that, across domains such as biology, physics, medicine, chemistry, and logic, the benefits of incorporating PDDL-derived data are even more pronounced. In these categories, PRMs trained on the combined corpus outperform their non-PDDL counterparts by a substantial margin, with the largest improvements again observed for the models fine-tuned on Math-Shepherd. Moreover, compared to the strongest public PRMs of comparable size, our PDDL-based models achieve superior performance, further highlighting the effectiveness of PDDL-derived supervision.

PDDL-Based Reasoning Evaluation Finally, when evaluated on the PDDL test set, the impact of PDDL2PRM integration becomes striking. PRMs trained on the augmented dataset achieve substantial F1 improvements, demonstrating a strong ability to assess planning-based reasoning. These gains also generalize to unseen scenarios: on the *Rooms* test set, performance improvements remain similarly pronounced. In contrast, other PRMs perform poorly on this logic-intensive task. Models not trained by us, as well as our Math-Shepherd-only variant, achieve substantially lower performance. Some such models exhibit a strong bias toward classifying most reasoning chains as correct, performing well on error-free samples but poorly on samples with errors. Others display the opposite behavior, predicting chains as incorrect while struggling to localize the first erroneous step (Table 6).

Impact on a Strong Pre-trained PRM Fine-tuning the PRM introduced by Zhang et al. (2025) with PDDL-derived data yields notable performance improvements. On mathematical benchmarks (Tables 3 and 4), performance remains stable, with slight improvements despite the non-mathematical nature of PDDL2PRM. In contrast, on non-mathematical benchmarks (Tables 5 and 6), the PRM achieves substantial gains, further demonstrating the effectiveness of the proposed dataset.

Qualitative Analysis In addition to the quantitative experiments, we conducted a qualitative analysis on a sample of 100 instances drawn from multiple benchmarks and models, with a view to better understanding how PDDL-derived data affects PRMs' ability to evaluate free-form natural language reasoning.

We observe that PRMs trained with PDDL-derived data are more effective at identifying the first incorrect step in a reasoning chain, showing greater sensitivity to early errors than those without PDDL supervision. This suggests a stronger ability to localize failures at their source, consistent with the quantitative results on the erroneous chains reported in Tables 3, 15, 16, and 17.

Importantly, our qualitative analysis suggests

Model	Math F1	Biology F1	Physics F1	Medicine F1	Chemistry F1	Logic F1	Avg. F1	Avg. F1 (no math)
Skywork-PRM-7B	21.0	27.1	23.1	10.2	27.1	13.1	20.3	20.1
Qwen2.5-Math-PRM-7B	55.5	22.8	36.4	23.5	39.5	19.0	32.8	28.2
†Qwen2.5-Math-PRM-7B-PDDL-r	**56.1**	29.6	36.2	26.7	**42.7**	24.0	35.9	31.8
Qwen2.5-Math-7B-PRM800k	47.7	17.7	28.7	18.8	33.2	8.8	25.8	21.4
†Qwen2.5-Math-7B-PRM800k-r	47.5	21.9	40.0	23.9	29.7	22.4	30.9	27.6
†Qwen2.5-Math-7B-PRM800k-PDDL-r	53.6	27.1	**45.0**	30.0	37.6	27.6	36.8	33.5
†Llama-3.1-8B-PRM800k-r	39.3	25.0	30.7	24.0	27.3	22.4	28.1	25.9
†Llama-3.1-8B-PRM800k-PDDL-r	44.4	**36.6**	42.1	**35.5**	36.4	**28.1**	**37.2**	**35.7**
†Qwen2.5-Math-7B-MathShepherd-r	25.8	2.8	8.3	2.1	9.4	0.6	8.2	4.6
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	43.2	13.3	33.6	9.0	20.6	15.6	22.5	18.4

Table 5: Results on MR-Ben. † indicates PRMs we trained; best results in **bold**. The full version of this table, including error and correct accuracies, is reported in Appendix F.

Model	PDDL test set			Rooms test set			Avg. F1
	Error	Correct	F1	Error	Correct	F1	
Skywork-PRM-7B	12.9	0.0	0.0	14.7	0.0	0.0	0.0
Qwen2.5-Math-PRM-7B	32.5	68.2	44.0	9.5	**100.0**	17.3	30.7
†Qwen2.5-Math-PRM-7B-PDDL-r	99.1	86.0	92.1	**83.1**	98.9	**90.3**	**91.2**
Qwen2.5-Math-7B-PRM800k	0.8	**100.0**	1.6	0.0	**100.0**	0.0	0.8
†Qwen2.5-Math-7B-PRM800k-r	39.7	87.5	54.6	14.8	**100.0**	25.7	40.2
†Qwen2.5-Math-7B-PRM800k-PDDL-r	**99.2**	83.7	90.8	75.4	86.8	80.7	85.8
†Llama-3.1-8B-PRM800k-r	49.4	51.1	50.2	46.9	99.5	63.7	57.0
†Llama-3.1-8B-PRM800k-PDDL-r	**99.2**	90.1	**94.5**	75.7	82.9	79.1	86.8
†Qwen2.5-Math-7B-MathShepherd-r	14.9	1.6	2.9	14.7	0.0	0.0	1.5
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	96.8	83.0	89.3	81.3	99.9	89.6	89.5

Table 6: Results on the PDDL test set. † indicates PRMs we trained; best results in **bold**.

that this gain is not limited to detecting arithmetic slips or local logical inconsistencies. Instead, PDDL-trained PRMs more reliably flag errors that result from an incorrect problem formulation, such as invalid assumptions, inappropriate reductions, or unjustified constraints that alter the structure of the task. In other words, they are better at catching modeling errors rather than only execution errors.

We hypothesize that this effect arises naturally from the planning formulation: PDDL makes preconditions and state transitions explicit, and training PRMs to judge whether an action violates formal constraints may encourage a verification style that transfers to natural language reasoning. Illustrative examples are provided in Appendix H.

Regression vs. Classification Notably, across all benchmarks, our model *Qwen2.5-Math-7B-PRM800k-r* consistently outperforms the one proposed by Zheng et al. (2025), *Qwen2.5-Math-7B-PRM800k*. Both models are fine-tuned from the same model and on the same dataset, differing only in the learning objective: regression versus classification. These results highlight the effectiveness of training PRMs as regressors when the supervision signal involves more than two distinct labels, as is the case for the PRM800k dataset. Additional experimental evidence is provided in Appendix D.

6 Conclusion

This work introduced a novel approach to PRM dataset construction leveraging PDDL domains as structured sources of reasoning data. We address key limitations of existing datasets, namely limited scalability due to high computational costs, narrow domain coverage, low granularity and precision. We propose a synthetic data generation framework that produces reliable step-level rewards beyond the mathematical domain and allows the resulting dataset to be easily expanded through newly created PDDL domains and problems. Evaluation across multiple benchmarks demonstrates that PRMs fine-tuned on a corpus augmented with PDDL-derived data consistently outperform their counterparts trained on the same starting data excluding the PDDL component. These results show that the proposed method and dataset effectively enhance the ability of PRMs to evaluate reasoning processes, highlighting PDDL-derived supervision as a reliable and promising resource for improving reasoning capabilities in LLMs.

Limitations

While our proposed approach provides several advantages, some limitations remain. First, the reward distribution is imbalanced: most examples receive reward values of 0.0, 0.5, or 1.0, while fewer instances fall into intermediate levels, such as 0.25 or 0.75. The 0.25 reward is intrinsically underrepresented, as several of the PDDL domains used are designed to remain solvable from almost any intermediate state, leaving very few actions that transition the system into an unsolvable configuration.

Second, the synthetic reasoning traces obtained by translating PDDL problems and actions into natural language lack the naturalness and fluency of CoTs produced by LLMs. Improving the linguistic quality of these translations is an important direction for future work, as it may enhance the alignment between synthetic supervision and the reasoning style of modern LLMs.

Finally, another promising direction involves the automatic translation of natural-language problems into PDDL domains. Such a method would enable the dynamic generation of structured reasoning tasks tailored to specific problem types, further expanding the applicability and effectiveness of PDDL-based supervision.

Acknowledgments

We gratefully acknowledge CINECA for providing the GPU computational resources, in particular through access to the HPC Leonardo infrastructure, which were essential for the development of this project.

References

Ella M. Atkins, Edmund H. Durfee, and Kang G. Shin. 1997. Detecting and reacting to unplanned-for world states. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*, pages 571–576. AAAI Press / The MIT Press.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*, abs/2503.09567.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, 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, Hao Peng, Yu Cheng, and 4 others. 2025. Process reinforcement through implicit rewards. *CoRR*, abs/2502.01456.

Rina Dechter and Judea Pearl. 1985. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32(3):505–536.

Maria Fox and Derek Long. 2002. The third international planning competition: Temporal and metric planning. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, page 333. AAAI.

Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2004. *Automated planning - theory and practice*. Elsevier.

Kai Goebel and Patrik Zips. 2025. Can llm-reasoning models replace classical planning? A benchmark study. *CoRR*, abs/2507.23589.

Google DeepMind. 2025. Gemini 3. https://deepmind.google/models/gemini/.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. 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*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. 2025. Skywork open reasoner 1 technical report. *CoRR*, abs/2505.22312.

Malte Helmert. 2006. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246.

Malte Helmert and Carmel Domshlak. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, 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. *CoRR*, abs/2406.06592.

Meta AI. 2024. Llama 3.1 8B Instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

Andrea Micheli, Arthur Bit-Monnot, Gabriele Röger, Enrico Scala, Alessandro Valentini, Luca Framba, Alberto Rovetta, Alessandro Trapasso, Luigi Bonassi, Alfonso Emilio Gerevini, Luca Iocchi, Félix Ingrand, Uwe Köckemann, Fabio Patrizi, Alessandro Saetti, Ivan Serina, and Sebastian Stock. 2025. Unified planning: Modeling, manipulating and solving AI planning problems in python. *SoftwareX*, 29:102012.

Mistral AI. 2025. Magistral. *CoRR*, abs/2506.10910.

Francesco Maria Molfese, Luca Moroni, Ciro Porcaro, Simone Conia, and Roberto Navigli. 2026. Retraceqa: Evaluating reasoning traces of small language models in commonsense question answering. In *Association for Computational Linguistics: ACL 2026*, San Diego, California. Association for Computational Linguistics.

OpenAI. 2024. Openai o1 system card. *CoRR*, abs/2412.16720.

James T. Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. 2024. Large language models as planning domain generators (student abstract). In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 23604–23605. AAAI Press.

Qwen Team. 2025a. Qwen2.5 Math 7B Instruct. https://huggingface.co/Qwen/Qwen2.5-Math-7B-Instruct.

Qwen Team. 2025b. Qwen3 Technical Report. *CoRR*, abs/2505.09388.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 25299–25346. Association for Computational Linguistics.

Kaya Stechly, Karthik Valmeekam, Atharva Gundawar, Vardhan Palod, and Subbarao Kambhampati. 2025a. Beyond semantics: The unreasonable effectiveness of reasonless intermediate tokens. *CoRR*, abs/2505.13775.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025b. On the self-verification limitations of large language models on reasoning and planning tasks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Wei Sun, Qianlong Du, Fuwei Cui, and Jiajun Zhang. 2025. An efficient and precise training data construction framework for process-supervised reward model in mathematical reasoning. *CoRR*, abs/2503.02382.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. 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*.

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.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo Hernandez, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. 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*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models - A critical investigation. 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*.

Pulkit Verma, Ngoc La, Anthony Favier, Swaroop Mishra, and Julie A. Shah. 2025. Teaching llms to plan: Logical chain-of-thought instruction tuning for symbolic planning. *CoRR*, abs/2509.13351.

Kaustubh Vyas, Damien Graux, Sébastien Montella, Pavlos Vougiouklis, Ruofei Lai, Keshuang Li, Yang Ren, and Jeff Z. Pan. 2025. An extensive evaluation of PDDL capabilities in off-the-shelf llms. *CoRR*, abs/2502.20175.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, Linyi Yang, Ying Wen, and Weinan Zhang. 2024a. Openr: An open source framework for advanced reasoning with large language models. *CoRR*, abs/2410.09671.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. 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.

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.

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, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

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, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Zhongshen Zeng, Yinhong Liu, Yingjia Wan, Jingyao Li, Pengguang Chen, Jianbo Dai, Yuxuan Yao, Rongwu Xu, Zehan Qi, Wanru Zhao, Linling Shen, Jianqiao Lu, Haochen Tan, Yukang Chen, Hao Zhang, Zhan Shi, Bailin Wang, Zhijiang Guo, and Jiaya Jia. 2024. Mr-ben: A meta-reasoning benchmark for evaluating system-2 thinking in llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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*, Findings of ACL, 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.

A Detailed Domain Descriptions

This appendix provides detailed descriptions of the PDDL domains used to generate our dataset.

A.1 BlocksWorld-4-ops

The BlocksWorld-4-ops domain models a classical block-stacking scenario in which a robotic arm manipulates a set of blocks arranged on a table. Each block can either rest on the table or on top of another block, and at most one block may be held by the arm at any given time.

State Representation The state of the environment is described using the following predicates:

- `on(x,y)`: block `x` is on top of block `y`;
- `on-table(x)`: block `x` is on the table;
- `clear(x)`: block `x` has no block on top of it;
- `holding(x)`: the arm is holding block `x`;
- `arm-empty`: the arm is not holding any block.

Actions The available actions are:

- `pick-up(x)`: the arm picks up block `x` from the table;
- `put-down(x)`: the arm places block `x` back on the table;
- `stack(x,y)`: the arm places block `x` on top of block `y`;
- `unstack(x,y)`: the arm removes block `x` from block `y`.

Problem Instances Each problem instance defines an initial arrangement of blocks and a target configuration that the agent must reproduce through a sequence of manipulation actions.

A.2 BlocksWorld-3-ops

The BlocksWorld-3-ops domain represents a simplified variant of the block-stacking scenario introduced in the previous section. Unlike its 4-operator counterpart, this domain does not model an explicit robotic arm: blocks are moved directly between locations through abstract manipulation actions.

State Representation Since no arm is present, the state is encoded using a reduced set of predicates:

- `on(x,y)`: block `x` is on top of block `y`;
- `on-table(x)`: block `x` is on the table;
- `clear(x)`: block `x` has no block on top of it.

Actions The available actions are:

- `move-b-to-t(x,y)`: block `x` is moved from on top of block `y` to the table;
- `move-t-to-b(x,y)`: block `x` is moved from the table onto block `y`;
- `move-b-to-b(x,y,z)`: block `x` is moved from on top of block `y` to on top of block `z`.

Problem Instances As in the 4-operator version, each problem instance defines an initial configuration of blocks and a target arrangement that must be reconstructed, but here the plan consists solely of abstract move operations rather than fine-grained arm manipulations.

A.3 Ferry

The Ferry domain models a transportation scenario in which a set of cars must be moved between different locations using a single ferry. The ferry can carry at most one car at a time, and its position determines where boarding and disembarking operations may occur.

State Representation The state is characterized by predicates that track the locations of both the ferry and the cars:

- `at-ferry(l)`: the ferry is currently at location `l`;
- `at(c,l)`: car `c` is at location `l`;
- `on(c)`: car `c` is loaded onto the ferry;
- `empty-ferry`: the ferry is not carrying a car.

Actions The available actions are:

- `sail(l1,l2)`: moves the ferry from location `l1` to location `l2`;
- `board(c,l)`: loads car `c` onto the ferry at location `l`;
- `debark(c,l)`: unloads car `c` from the ferry at location `l`.

Problem Instances Each problem instance specifies the initial locations of the cars and the ferry, together with a target configuration requiring specific cars to be delivered to designated locations.

A.4 Tower of Hanoi

The Tower of Hanoi domain models the classical puzzle in which disks of different sizes are stacked on pegs and must be moved to reach a target configuration while preserving the size ordering constraint.

State Representation The state of the environment is described using the following predicates:

- `clear(x)`: no disk is on top of `x`;
- `on(x,y)`: disk `x` is on top of `y` (which can be either another disk or a peg);
- `smaller(x,y)`: disk `x` is smaller than `y`.

Actions The available action is:

- `move(d,f,t)`: the disk `d` is moved from support `f` to support `t`.

Problem Instances Each problem instance defines an initial stacking of disks across the available pegs and a target configuration.

A.5 Logistics

The Logistics domain models a transportation scenario in which packages, trucks, and airplanes must be routed across different locations to achieve a specified delivery configuration. The environment is organized into cities, each containing several locations, and only airports allow inter-city movement.

State Representation The state of the environment is described using the following predicates:

- `at(x,l)`: object or vehicle `x` is at location `l`;
- `in(x,y)`: object `x` is loaded inside vehicle `y`;
- `in-city(l,c)`: location `l` belongs to city `c`;

Actions The available actions are:

- `load-truck(o,t,l)`: loads the object `o` into the truck `t` when both are at location `l`;
- `unload-truck(o,t,l)`: unloads the object `o` from truck `t` at location `l`;
- `load-airplane(o,a,l)`: loads the object `o` into the airplane `a` when both are at location `l`;
- `unload-airplane(o,a,l)`: unloads the object `o` from the airplane `a` at location `l`;
- `drive-truck(t,l1,l2,c)`: moves the truck `t` from location `l1` to location `l2` within the city `c`;
- `fly-airplane(a,l1,l2)`: flies the airplane `a` from location `l1` to location `l2`.

Problem Instances Each problem instance specifies the initial locations of all packages and vehicles, together with a target configuration that requires delivering the packages to their designated destinations.

A.6 Elevator

The Elevator domain (also known as Miconic) models a passenger transportation scenario in which an elevator must move across floors to pick up and drop off passengers according to their origin and destination floors.

State Representation The state of the environment is described using the following predicates:

- `origin(p,f)`: passenger `p` is initially located at floor `f`;
- `destin(p,f)`: floor `f` is the destination of passenger `p`;
- `above(f1,f2)`: floor `f1` is above floor `f2`;
- `boarded(p)`: passenger `p` is inside the elevator;
- `served(p)`: passenger `p` has reached their destination;
- `lift-at(f)`: the elevator is at floor `f`.

Actions The available actions are:

- `board(f,p)`: passenger `p` boards the elevator at floor `f`;
- `depart(f,p)`: passenger `p` exits the elevator at floor `f`;
- `up(f1,f2)`: the elevator moves upward from floor `f1` to floor `f2`;
- `down(f1,f2)`: the elevator moves downward from floor `f1` to floor `f2`;

Problem Instances Each problem instance specifies the origin and destination floors of all passengers, the initial position of the elevator, and requires transporting every passenger to their assigned destination.

A.7 N-Puzzle

The N-Puzzle domain models a sliding-tile puzzle in which numbered tiles must be rearranged on a grid to reach a target configuration, using a single empty position to move tiles.

State Representation The state of the environment is described using the following predicates:

- `at(t,p)`: tile `t` is at position `p`;
- `empty(p)`: position `p` is currently empty.

Actions The available action is:

- `move(t,p1,p2)`: tile `t` is moved from position `p1` to position `p2`.

Problem Instances Each problem instance defines an initial placement of all tiles and the empty position on the grid, together with a target configuration that must be reached.

A.8 VisitGrid

The VisitGrid domain models a navigation task in which a robot must traverse a grid-like environment and visit a predefined set of locations.

State Representation The state of the environment is described using the following predicates:

- `at-robot(l)`: the robot is currently located at location `l`;
- `visited(l)`: location `l` has been visited.

Actions The available action is:

- `move(l1,l2)`: moves the robot from location `l1` to location `l2`.

Problem Instances Each problem instance specifies the robot's initial position and a set of target locations that must be visited at least once.

A.9 Sokoban

The Sokoban domain models a grid-based puzzle in which a robot must navigate through an environment with boxes and obstacles, pushing the boxes to new locations to reach a desired configuration.

State Representation The state of the environment is described using the following predicates:

- `at-robot(l)`: the robot is at location `l`;
- `at(b,l)`: box `b` is at location `l`;
- `clear(l)`: location `l` does not contain a box.

Actions The available actions are:

- `move(l1,l2,dir)`: the robot moves from location `l1` to location `l2`;
- `push(l1,l2,b)`: the robot pushes box `b` from location `l1` to location `l2`.

Problem Instances Each problem instance specifies the initial positions of the robot and all boxes, together with a target configuration that requires placing each box in a designated location.

A.10 Spanner

The Spanner domain models a maintenance task in which an agent must move across a set of locations, collect spanners with limited durability, and use them to tighten a set of loose nuts distributed throughout the environment.

State Representation The state of the environment is described using the following predicates:

- `at(x,l)`: object or agent `x` is at location `l`;
- `loose(n)`: nut `n` is loose;
- `tightened(n)`: nut `n` has been tightened;
- `useable1(s)`, `useable2(s)`: spanner `s` has one or two remaining uses;
- `link(l1,l2)`: location `l1` is connected to location `l2`.

Actions The available actions are:

- `move(actor,l1,l2)`: the agent moves from location `l1` to location `l2`;
- `pick-up(actor,s)`: the agent picks up spanner `s`;
- `tighten(actor,n,s)`: the agent tightens nut `n` using spanner `s`, consuming one unit of its durability.

Problem Instances Each problem instance defines the initial location of the agent, the placement and durability of the available spanners, and the set of loose nuts that must be tightened.

A.11 Rooms

The Rooms domain models a navigation task in which an agent moves across a collection of rooms connected by fragile doors and must turn off all lights in the environment before losing access to certain areas.

State Representation The state of the environment is described using the following predicates:

- `at(a,r)`: the agent `a` is located in room `r`;
- `door(r1,r2)`: rooms `r1` and `r2` are connected by a door;
- `door-intact(r1,r2)`: the door between `r1` and `r2` is intact and can still be traversed;
- `on(r)`: the light in room `r` is switched on.

Actions The available actions are:

- `move(a,r1,r2)`: the agent moves from room `r1` to room `r2` breaking the traversal door;
- `turn-off(a,r)`: the agent turns off the light in room `r`.

Problem Instances Each problem instance specifies the initial position of the agent, the layout of the rooms and doors, and the subset of rooms whose lights are initially on and must be turned off.

B Training Implementation Details

The architecture of the PRM was built using a head added on top of the decoder: the head maps the decoder's final-layer hidden states to a scalar score for every token in the sequence. Concretely, it consists of two linear layers with an intermediate Tanh activation (Hidden $\rightarrow$ Hidden/2 $\rightarrow$ 1). During the forward pass the computation proceeds as follows:

1. The input sequence is fed to the base decoder, which returns last-layer hidden states.
2. The head is applied token-wise to them, producing raw scalar logits (one per token).

The prompt, before being passed to the model, is enriched with special marker tokens that explicitly indicate the positions where a reward prediction is required (inserted immediately after each reasoning step, as done by Wang et al. (2024a)). Only these marker positions are relevant for the supervision: the trainer constructs a boolean mask that is 1 at marker positions and 0 elsewhere, and selects the corresponding predictions and targets. Only predictions at marker positions contribute to the loss. When trained in the regression setting, the model minimizes the Mean Squared Error (MSE) between predicted rewards and target scores. In the classification setting, we instead optimize the Binary Cross-Entropy (BCE) loss with logits.

Fine-tuning was conducted using Low-Rank Adaptation (LoRA) (Hu et al., 2022). Both the LoRA adapters and the head weights are updated, while the remaining model parameters remain frozen. The parameters used for LoRA are shown in Table 7.

Parameter	Value
Rank	64
LoRA scaling factor (α)	64
Target modules	Q, K, V, O of the transformer + main feed-forward layers
LoRA dropout	0.1

Table 7: LoRA configuration.

All models were fine-tuned using FP16 precision. The learning rate was selected in the range $[1 \times 10^{-5}, 3 \times 10^{-5}]$ and adjusted based on the size of the training dataset. A batch size of 32 and a weight decay of 0.05 were used in all experiments.

The PRM800k dataset was pre-processed to match the structure of the PDDL-derived dataset and the Math-Shepherd dataset, including the removal of incomplete samples and entries lacking step-level reward annotations. Each resulting sample consists of a problem description and a (possibly partial) CoT comprising a variable number of reasoning steps, along with the corresponding step-level rewards. After filtering, the processed PRM800k dataset contains approximately 350k samples. On the other hand, the Math-Shepherd dataset comprises approximately 400k samples. To ensure a balanced training process when combining PDDL2PRM with either PRM800k or Math-Shepherd, the PDDL-derived dataset was not used in its entirety. Instead, we randomly sampled a subset of instances from the PDDL2PRM training corpus whose size matches that of the paired dataset used during training.

C Ablation Study: PDDL-Only SFT

In this appendix, we report results for a model trained exclusively on PDDL-derived supervision. The architecture, regression-based training setup, and optimization procedure are identical to those used in the main experiments. The only difference lies in the training data, which consists solely of the PDDL-derived dataset that is used as part of the mixed supervision for the other models.

As shown in Tables 8, 9, and 10, the model trained exclusively on PDDL-derived data underperforms models trained with natural-language reasoning supervision, such as PRM800k or Math-Shepherd, across the benchmarks. This performance gap is particularly pronounced on the mathematical evaluations of ProcessBench, which fall outside the scope of PDDL-based supervision. An exception is MR-Ben, where the PDDL-trained model outperforms the Math-Shepherd–trained model, as well as the PDDL reasoning evaluation, on which the PDDL-trained model achieves superior performance.

D Regression vs. Classification Detailed Results

To further investigate the differences between regression- and classification-based training, we additionally trained two classification-based PRMs. Specifically, we trained one model starting from *Llama-3.1-8B-Instruct* on PRM800k by mapping neutral labels to the positive class, following the procedure adopted by Zheng et al. (2025), and another model starting from *Qwen2.5-Math-7B-Instruct* on Math-Shepherd to enable a direct comparison with the regression-trained PRMs presented earlier.

Tables 11, 12, and 13 report a detailed comparison between regression- and classification-based PRM training. For PRM800k, regression consistently outperforms classification across all benchmarks, as the dataset provides three distinct supervision levels that are more naturally captured by a regression objective. In contrast, for Math-Shepherd the results are nearly identical, with a slight advantage for the classification setting; this outcome is expected, since the dataset inherently contains only two labels. Overall, these results confirm that regression losses are more suitable when supervision involves multiple graded labels.

E PRMBench Detailed Results

Tables 14 and 15 provide a detailed breakdown of the results on the PRMBench benchmark. In particular, *Positive Acc* and *Negative Acc* denote accuracy on positive and negative reasoning steps, respectively, while *First* measures accuracy in identifying the first erroneous step in the CoT. *Similarity* is a measure of a model's ability to distinguish between correct and incorrect reasoning steps. *Positive F1* and *Negative F1* report the F1 scores on positive and negative steps, respectively.

In addition, PRMBench reports several diagnostic metrics:

- **NR (Non-Redundancy):** evaluates the PRM's ability to identify redundant steps within the reasoning process.
- **NCL (Non-Circular Logic):** assesses the PRM's ability to detect circular reasoning.
- **ES (Empirical Soundness):** measures the ability to detect counterfactual errors within the reasoning process.
- **SC (Step Consistency):** evaluates whether the PRM can detect step-wise contradictions.
- **DC (Domain Consistency):** measures the ability to identify domain-inconsistent reasoning, a specific type of counterfactual error.
- **CI (Confidence Invariance):** evaluates whether the PRM can detect over-confident errors.
- **PS (Prerequisite Sensitivity):** measures sensitivity to missing conditions or prerequisite violations.
- **DR (Deception Resistance):** evaluates the ability to detect deceptive or misleading reasoning steps.
- **MS (Multi-Solution Consistency):** assesses consistency across different solution paths for the same problem.

F MR-Ben Detailed Results

Tables 16 and 17 report the complete results on MR-Ben, including accuracy on fully correct CoTs as well as accuracy on chains containing reasoning errors.

G Computational Costs

We report the computational resources required to train the PRMs used in this work. All models were trained using A100 GPUs. For each training run, we used 8 GPUs in parallel.

Training time scales approximately linearly with the size of the training corpus. For models trained on the PRM800k dataset or on the Math-Shepherd dataset alone, training required between 4 and 6 hours, corresponding to approximately 32–48 GPU-hours per model. Models trained on the combined corpus required between 7 and 10 hours, corresponding to approximately 56–80 GPU-hours.

Training was performed using standard distributed data-parallel setups. Limited hyperparameter tuning was performed to adjust key training parameters, including the learning rate, weight decay, and LoRA dropout.

H Additional Qualitative Examples

To further illustrate the qualitative pattern discussed in Section 5, we report two representative examples in Figures 2 and 3. In both cases, the PRM trained with PDDL-derived supervision more reliably penalizes reasoning steps that introduce an incorrect formulation of the problem itself. In Figure 2, the chain begins by imposing an invalid combinatorial structure on the task, collapsing the space of valid arrangements into three fixed groups and thereby excluding legitimate alternating patterns.

In Figure 3, the reasoning introduces a biologically invalid conceptual statement, which establishes incorrect foundations for the subsequent argument.

In both examples, this difference is reflected in the reward trajectories: the PRM trained without PDDL supervision continues to assign relatively high scores even after the incorrect formulation has been introduced, whereas the PRM trained with PDDL supervision drops more sharply once the chain departs from a valid problem representation.

I Licenses

In this work we leverage several existing datasets and pretrained models. Below we summarize their licensing terms in accordance with the information provided by the original authors and public repositories:

- **PRM800k**: Dataset released under the MIT License.
- **Math-Shepherd**: Publicly available dataset released without an explicit license. It is used strictly for research and experimental purposes.
- **Skywork-o1-Open-PRM-Qwen-2.5-7B**: Released under the Skywork Community License, that permits research and experimental use.
- **Qwen2.5-Math models**: Released under the Apache License, Version 2.0.
- **meta-llama/Llama-3.1-8B-Instruct**: Distributed under the Meta LLaMA 3.1 Community License, which permits research and experimental use.

The dataset and trained models introduced in this work will be released under the Creative Commons Attribution-ShareAlike (CC BY-SA 4.0) license.

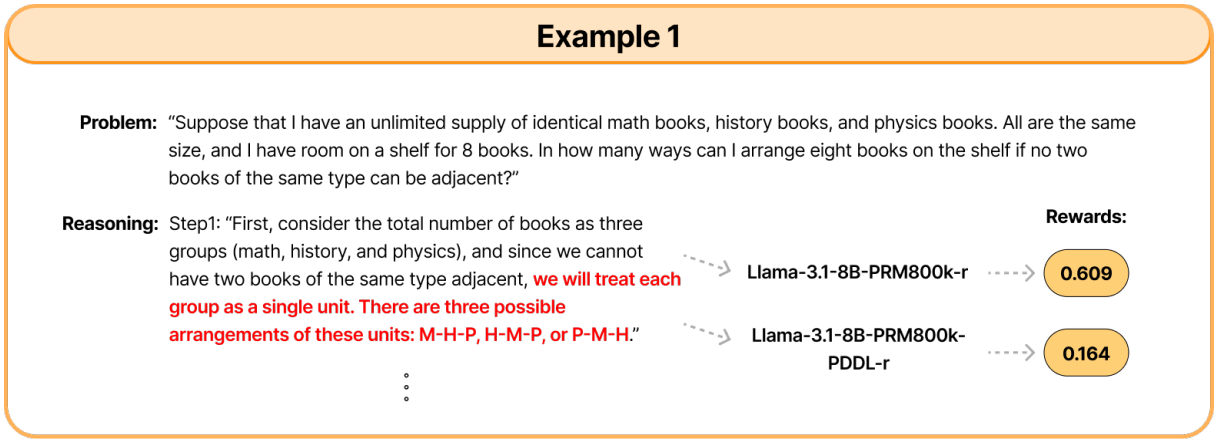

Figure 2: Example from the MATH subset of ProcessBench. We report scores from two PRMs based on Llama + PRM800k. The PDDL-trained PRM more strongly penalizes the incorrect combinatorial formulation, where the reasoning restricts the solution space to permutations of fixed groups, excluding valid interleavings across types.

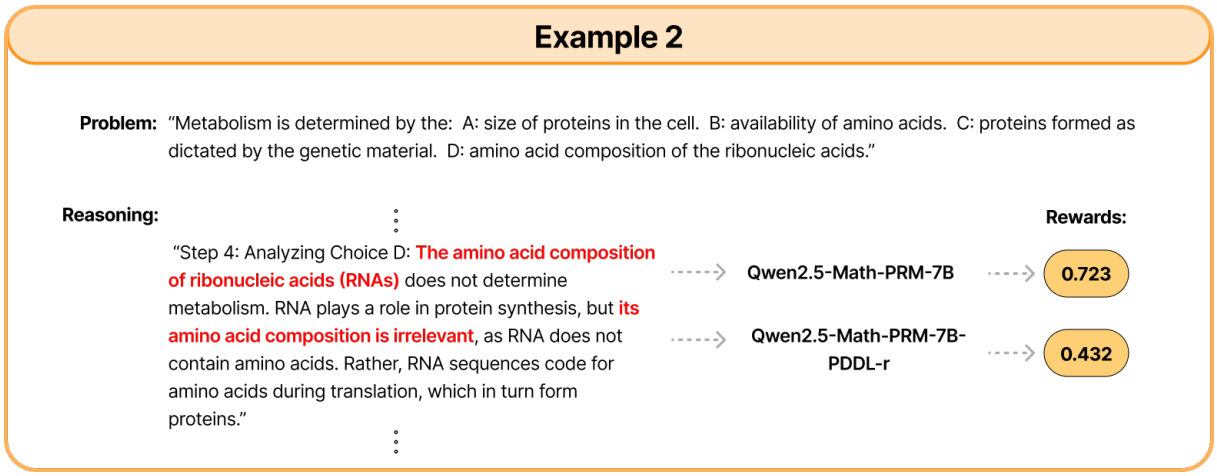

Figure 3: Example from the Medicine subset of MR-Ben. We report scores from two PRMs based on Qwen2.5-Math-PRM-7B. The PDDL-trained PRM assigns a lower score to the fourth reasoning step, correctly penalizing a biologically incorrect justification that conflates RNA composition with its functional role in protein synthesis.

Model	GSM8K			MATH			OlympiadBench			Omni-MATH			Avg. F1
	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	
Qwen2.5-Math-7B-MathShepherd-r	51.7	96.9	67.4	20.0	96.8	33.2	7.3	95.9	13.5	4.0	96.7	7.6	30.4
Qwen2.5-Math-7B-PRM800k-r	58.0	89.6	70.4	60.8	74.4	66.9	48.4	56.0	52.0	49.7	57.3	53.2	60.6
Qwen2.5-Math-7B-PDDL-r	33.3	9.3	14.6	24.4	22.7	23.5	15.4	35.1	21.4	9.2	28.6	14.0	18.4

Table 8: Results of the PDDL-only model on ProcessBench compared with MathShepherd and PRM800k models.

Model	Math F1	Biology F1	Physics F1	Medicine F1	Chemistry F1	Logic F1	Avg. F1	Avg. F1 (no math)
Qwen2.5-Math-7B-MathShepherd-r	25.8	2.8	8.3	2.1	9.4	0.6	8.2	4.6
Qwen2.5-Math-7B-PRM800k-r	47.5	21.9	40.0	23.9	29.7	22.4	30.9	27.6
Qwen2.5-Math-7B-PDDL-r	26.3	7.8	21.8	11.3	12.5	10.5	15.0	12.8

Table 9: Results of the PDDL-only model with Math-Shepherd and PRM800k models on the MR-Ben benchmark.

Model	PDDL test set			Rooms test set			Avg. F1
	Error	Correct	F1	Error	Correct	F1	
Qwen2.5-Math-7B-MathShepherd-r	14.9	1.6	2.9	14.7	0.0	0.0	1.5
Qwen2.5-Math-7B-PRM800k-r	39.7	87.5	54.6	14.8	100.0	25.7	40.2
Qwen2.5-Math-7B-PDDL-r	98.9	84.9	91.3	78.3	90.6	84.0	87.7

Table 10: Performance of the PDDL-only model on the PDDL and Rooms test sets, compared with models trained on Math-Shepherd and PRM800k.

Model	GSM8K			MATH			OlympiadBench			Omni-MATH			Avg. F1
	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1	
Qwen2.5-Math-7B-PRM800k	53.1	95.3	68.2	48.0	90.1	62.6	35.7	87.3	50.7	29.8	86.1	44.3	56.5
†Qwen2.5-Math-7B-PRM800k-r	**58.0**	89.6	**70.4**	**60.8**	74.4	**66.9**	**48.4**	56.0	**52.0**	**49.7**	57.3	**53.2**	**60.6**
†Llama-3.1-8B-PRM800k-c	39.1	89.1	54.4	43.4	58.1	49.7	40.7	35.4	37.9	39.9	38.6	39.2	45.3
†Llama-3.1-8B-PRM800k-r	45.9	91.2	61.1	46.0	58.1	51.3	43.7	31.3	36.5	42.6	35.3	38.6	46.9
†Qwen2.5-Math-7B-MathShepherd-c	50.2	**97.4**	66.3	19.9	96.6	33.0	7.4	**96.2**	13.8	4.9	95.9	9.3	30.6
†Qwen2.5-Math-7B-MathShepherd-r	51.7	96.9	67.4	20.0	**96.8**	33.2	7.3	95.9	13.5	4.0	**96.7**	7.6	30.4

Table 11: Classification vs. Regression on ProcessBench. † indicates PRMs we trained; best results in **bold**. (Qwen2.5-Math-7B-PRM800k is trained as a classifier).

Model	Math F1	Biology F1	Physics F1	Medicine F1	Chemistry F1	Logic F1	Avg. F1	Avg. F1 (no math)
Qwen2.5-Math-7B-PRM800k	**47.7**	17.7	28.7	18.8	**33.2**	8.8	25.8	21.4
†Qwen2.5-Math-7B-PRM800k-r	47.5	21.9	**40.0**	23.9	29.7	**22.4**	**30.9**	**27.6**
†Llama-3.1-8B-PRM800k-c	35.0	20.6	33.4	**25.8**	26.4	20.1	26.9	25.3
†Llama-3.1-8B-PRM800k-r	39.3	**25.0**	30.7	24.0	27.3	**22.4**	28.1	25.9
†Qwen2.5-Math-7B-MathShepherd-c	25.9	2.8	7.7	1.7	9.8	1.4	8.2	4.7
†Qwen2.5-Math-7B-MathShepherd-r	25.8	2.8	8.3	2.1	9.4	0.6	8.2	4.6

Table 12: Classification vs. Regression on MR-Ben. † indicates PRMs we trained; best results in **bold**. (Qwen2.5-Math-7B-PRM800k is trained as a classifier).

Model	PDDL test set			Rooms test set			Avg. F1
	Error	Correct	F1	Error	Correct	F1	
Qwen2.5-Math-7B-PRM800k	0.8	**100.0**	1.6	0.0	**100.0**	0.0	0.8
†Qwen2.5-Math-7B-PRM800k-r	39.7	87.5	**54.6**	14.8	**100.0**	25.7	40.2
†Llama-3.1-8B-PRM800k-c	**51.0**	34.8	41.4	45.4	99.4	62.3	51.9
†Llama-3.1-8B-PRM800k-r	49.4	51.1	50.2	**46.9**	99.5	**63.7**	**57.0**
†Qwen2.5-Math-7B-MathShepherd-c	14.8	1.0	1.9	14.8	0.1	0.1	1.0
†Qwen2.5-Math-7B-MathShepherd-r	14.9	1.6	2.9	14.7	0.0	0.0	1.5

Table 13: Classification vs. Regression on PDDL test set. † indicates PRMs we trained; best results in **bold**. (Qwen2.5-Math-7B-PRM800k is trained as a classifier).

Model	Simplicity			Soundness					Sensitivity				Overall
	NR.	NCL.	Avg.	ES	SC.	DC.	CI	Avg.	PS	DR.	MS.	Avg.	
Skywork-PRM-7B	56.4	**62.8**	59.6	69.4	67.1	67.7	69.9	68.5	**60.9**	65.8	93.2	73.3	65.1
†Qwen2.5-Math-7B-PDDL-r	**59.1**	60.8	**59.9**	63.3	62.0	62.5	65.9	63.4	59.2	58.7	42.5	53.5	61.4
Qwen2.5-Math-PRM-7B	49.0	55.1	52.1	71.8	67.3	66.3	78.5	71.0	57.6	69.1	**99.7**	**75.5**	65.5
†Qwen2.5-Math-PRM-7B-PDDL-r	50.6	60.1	55.3	**73.7**	**69.1**	**67.8**	**78.6**	**72.3**	60.2	**70.5**	49.9	60.2	**67.5**
†Qwen2.5-Math-7B-MathShepherd-c	47.4	49.5	48.5	56.3	51.6	51.3	57.5	54.2	51.9	56.3	46.9	51.7	53.4
†Qwen2.5-Math-7B-MathShepherd-r	46.7	50.2	48.5	57.5	52.6	51.4	59.0	55.1	51.7	56.8	47.2	51.9	54.0
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	48.8	55.7	52.3	66.7	60.7	60.4	67.7	63.8	56.8	62.9	49.7	56.5	61.0

Table 14: Simplicity, Soundness and Sensitivity results on PRMBench. † indicates PRMs we trained; best results in **bold**.

Model	Total Acc	Positive Acc	Negative Acc	First	Similarity	Positive F1	Negative F1	PRMScore
Skywork-PRM-7B	81.7	88.5	42.7	**56.6**	90.1	89.2	40.9	65.1
†Qwen2.5-Math-7B-PDDL-r	75.9	81.4	**46.2**	53.3	84.6	85.0	37.7	61.4
Qwen2.5-Math-PRM-7B	**85.1**	**95.4**	30.6	37.8	89.0	**91.5**	39.4	65.5
†Qwen2.5-Math-PRM-7B-PDDL-r	84.4	92.8	39.2	47.4	86.3	90.9	**44.2**	**67.5**
†Qwen2.5-Math-7B-MathShepherd-c	68.6	74.6	36.5	33.1	84.4	80.0	26.8	53.4
†Qwen2.5-Math-7B-MathShepherd-r	70.4	77.3	33.9	30.5	84.6	81.5	26.5	54.0
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	78.0	85.6	37.9	38.9	**84.0**	86.8	35.3	61.0

Table 15: Detailed results on PRMBench. † indicates PRMs we trained; best results in **bold**.

Model	Math			Biology			Physics		
	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1
Skywork-PRM-7B	28.1	16.8	21.0	**26.6**	27.7	27.1	23.0	23.2	23.1
Qwen2.5-Math-PRM-7B	45.5	71.1	55.5	13.2	86.1	22.8	24.4	71.8	36.4
†Qwen2.5-Math-PRM-7B-PDDL-r	**48.9**	65.8	**56.1**	18.4	75.7	**29.6**	25.3	63.6	36.2
Qwen2.5-Math-7B-PRM800k	36.8	67.8	47.7	9.8	90.3	17.7	18.1	69.3	28.7
†Qwen2.5-Math-7B-PRM800k-r	37.8	63.8	47.5	13.6	56.2	21.9	30.7	57.4	40.0
†Qwen2.5-Math-7B-PRM800k-PDDL-r	47.3	61.7	53.6	17.7	57.7	27.1	**39.7**	52.0	**45.0**
†Qwen2.5-Math-7B-PDDL-r	20.7	36.2	26.3	4.1	83.3	7.8	13.2	61.8	21.8
†Llama-3.1-8B-PRM800k-c	23.5	68.5	35.0	11.7	85.7	20.6	22.4	65.8	33.4
†Llama-3.1-8B-PRM800k-r	27.4	69.1	39.3	14.6	87.2	25.0	20.1	64.9	30.7
†Llama-3.1-8B-PRM800k-PDDL-r	39.8	50.3	44.4	24.6	71.5	36.6	33.6	56.4	42.1
†Qwen2.5-Math-7B-MathShepherd-c	15.2	86.6	25.9	1.4	99.0	2.8	4.0	97.8	7.7
†Qwen2.5-Math-7B-MathShepherd-r	15.1	**87.9**	25.8	1.4	**99.5**	2.8	4.3	**98.1**	8.3
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	31.9	67.1	43.2	7.2	94.0	13.3	21.0	85.0	33.6

Table 16: Complete results on MR-Ben (Math, Biology and Physics domains). † indicates PRMs we trained; best results in **bold**.

Model	Medicine			Chemistry			Logic		
	Error	Correct	F1	Error	Correct	F1	Error	Correct	F1
Skywork-PRM-7B	19.1	7.0	10.2	29.0	25.4	27.1	**26.6**	8.7	13.1
Qwen2.5-Math-PRM-7B	14.0	73.2	23.5	26.8	75.1	39.5	10.9	73.1	19.0
†Qwen2.5-Math-PRM-7B-PDDL-r	16.8	64.6	26.7	**31.5**	66.2	**42.7**	14.9	60.9	24.0
Qwen2.5-Math-7B-PRM800k	10.5	88.7	18.8	21.5	73.6	33.2	4.7	77.9	8.8
†Qwen2.5-Math-7B-PRM800k-r	15.4	53.7	23.9	20.2	56.3	29.7	14.6	47.8	22.4
†Qwen2.5-Math-7B-PRM800k-PDDL-r	23.8	40.5	**30.0**	29.4	52.1	37.6	21.4	38.8	**27.6**
†Qwen2.5-Math-7B-PDDL-r	6.1	72.8	11.3	6.9	70.2	12.5	5.7	66.6	10.5
†Llama-3.1-8B-PRM800k-c	15.8	70.8	25.8	16.1	74.1	26.4	11.6	74.1	20.1
†Llama-3.1-8B-PRM800k-r	14.0	84.4	24.0	16.7	73.6	27.3	13.4	69.5	22.4
†Llama-3.1-8B-PRM800k-PDDL-r	**24.9**	61.9	35.5	26.4	58.6	36.4	19.3	51.3	28.1
†Qwen2.5-Math-7B-MathShepherd-c	0.9	97.7	1.7	5.2	**96.9**	9.8	0.7	98.2	1.4
†Qwen2.5-Math-7B-MathShepherd-r	1.1	**98.8**	2.1	4.9	96.3	9.4	0.3	**99.5**	0.6
†Qwen2.5-Math-7B-MathShepherd-PDDL-r	4.7	89.5	9.0	11.6	90.8	20.6	8.7	77.1	15.6

Table 17: Complete results on MR-Ben (Medicine, Chemistry and Logic domains). † indicates PRMs we trained; best results in **bold**.