

Process and Outcome Reward Modeling for Reasoning in Small Language Models

1st Nathan Gamache

Department of Mathematics and Computer Science
Lawrence Technological University
Southfield, Michigan, USA
ngamache@ltu.edu

2nd Sean Brainard

Department of Mathematics and Computer Science
Lawrence Technological University
Southfield, Michigan, USA
sbrainard@ltu.edu

Abstract—Recent work on process supervision has shown that language models can be guided toward more reliable reasoning by training reward models that evaluate intermediate steps instead of only final answers [1]. Most existing systems, however, focus on frontier-scale models and very large proprietary datasets [3], [6]. In this project we study reward modeling in a more constrained but practically important setting: a self-hosted small language model (SLM) with only a few billion parameters. We construct an end-to-end pipeline that (i) generates chain-of-thought (CoT) solutions from a 3B-parameter SLM, (ii) uses ChatGPT as a teacher model to label step-level and outcome-level correctness, and (iii) fine-tunes two 1.5B-parameter reward models on top of this backbone: a process reward model (PRM) that scores each reasoning step and an outcome reward model (ORM) that scores full solutions. At inference time we use these reward models to select among multiple CoT candidates. On a synthetic but non-trivial set of math and logic questions, our PRM achieves over 98% step-level accuracy and enables more reliable selection of CoT traces than simple majority vote on final answers. We discuss how these reward models could be used as the foundation for future reinforcement learning, in particular proximal policy optimization (PPO) in the style of RLHF and DeepSeek-R1 [4], [7], to further improve SLM reasoning without relying on closed-source mega-scale models at deployment time.

Index Terms—large language models, small language models, chain-of-thought reasoning, reward modeling, process supervision, outcome supervision, reinforcement learning

I. INTRODUCTION

Large language models (LLMs) such as GPT-3, 4, 5 and production systems like ChatGPT, Gemini, and Claude have demonstrated strong performance on math, logic, and programming tasks when prompted to think step by step [2], [9]. Chain-of-thought (CoT) prompting encourages the model to write out intermediate reasoning before committing to a final answer and often improves accuracy on multi-step problems [2]. However, CoT does not guarantee correctness: models frequently produce fluent but flawed derivations that culminate in wrong answers. For safety-critical or education-oriented applications, it is not enough to see reasoning; we must also evaluate whether that reasoning is correct.

Recent work proposes training *reward models* that judge the quality of reasoning traces and then using those rewards for selection or reinforcement learning [1], [3]–[5]. Lightman et al. distinguish between *outcome supervision*, which labels only final answers, and *process supervision*, which labels

each intermediate step [1]. Their process reward model improves over outcome-only selection on mathematical reasoning benchmarks, but relies on large proprietary teacher models and large-scale datasets (PRM800K).

At the same time, there is growing interest in *small language models* (SLMs), which are open models in the 1 to 7B parameter range that can be fully self-hosted and fine-tuned [8], [10]. SLMs offer practical advantages in accessibility and controllability, but lag behind frontier LLMs in raw reasoning ability. An open question is how far reward modeling alone can push an SLM toward more reliable reasoning when paired with a strong but external teacher model such as GPT-4o mini.

Goal and Contributions

Our central research question is: *Can process and outcome reward models learned from ChatGPT labels significantly improve the reasoning behavior of a self-hosted small language model?*

This paper makes the following contributions:

- We design a teacher–student pipeline in which ChatGPT labels chain-of-thought solutions generated by a 3B-parameter SLM, producing paired datasets for step-level (PRM) and outcome-level (ORM) supervision [1].
- We fine-tune two 1.5B-parameter reward models, namely a PRM over individual steps and an ORM over full solutions, and integrate them into several inference-time selection strategies for multi-step reasoning.
- We empirically evaluate these models on math and logic questions, showing that the PRM achieves strong step-level accuracy and that PRM-based selection offers qualitative improvements over majority vote and single-pass CoT on our synthetic benchmarks.
- We outline a reinforcement learning plan based on PPO that would use PRM and ORM scores as reward signals and discuss how such an extension could further improve SLM reasoning.

The remainder of this paper is organized as follows. Section II reviews related work on CoT prompting, reward modeling, RLHF, and SLMs. Section III describes our teacher–student pipeline, prompting strategy, reward model architectures, and selection algorithm. Section IV details the experimental setup, baselines, and implementation. Section V

presents results and discussion. Section VI concludes with limitations, future work, and broader impact.

II. RELATED WORK

A. Chain-of-Thought Reasoning

Chain-of-thought prompting elicits intermediate reasoning steps by asking the model to think step by step or by providing exemplars with worked solutions [2]. This approach improves performance on tasks such as GSM8K, symbolic reasoning, and logical puzzles, especially for larger models [9]. Self-consistency further improves results by sampling multiple CoT traces and selecting the most frequent final answer [2]. However, if many traces share the same subtle error, majority vote amplifies that error rather than correcting it.

Our work uses CoT sampling as a building block, but adds explicit process and outcome reward models to score reasoning quality. This shifts the focus from only counting final answers to evaluating the internal structure of the derivation.

B. Reward Modeling and RLHF

Reinforcement learning from human feedback (RLHF) trains a reward model on human preference data and then optimizes a policy model with PPO or related methods [3], [4]. Most RLHF pipelines supervise final outputs such as helpfulness or harmlessness [6]. Lightman et al. show that process supervision, which labels intermediate steps, can be more effective for math reasoning tasks than outcome-only supervision [1]. Direct Preference Optimization (DPO) further simplifies preference-based training by treating the language model as an implicit reward model [5].

We adopt the process and outcome supervision idea, but focus on a small open model and a small teacher-labeled dataset. We restrict ourselves to inference-time selection and do not train the generator with RL, which keeps the system simple and easier to reproduce on modest hardware.

C. Small Language Models

Open SLMs such as LLaMA 2 and Qwen-family models provide controllable backbones that can be fully hosted and modified by end users [8], [10]. They are attractive for research and deployment because of lower compute and memory requirements, at the cost of weaker zero-shot performance compared to frontier models. Our work fits into this line by using a 3B-parameter SLM as the generator and 1.5B-parameter SLMs as reward model bases.

Compared to prior work on process supervision, which targets larger closed models and large-scale datasets [1], our contribution is to show that similar ideas can be implemented in a fully self-hosted small-model setting and still provide meaningful gains in reasoning reliability.

III. METHODOLOGY

A. Task and Data

We focus on short-answer math and logic questions that require multi-step reasoning. Each question q is associated

with a domain label (*math* or *logic*) and is stored in a JSON Lines file with fields: *id*, *domain*, and *question*.

For each q , we generate N chain-of-thought candidate solutions using a 3B-parameter instruction-tuned SLM. Sampling uses the CoT prompt described in Section III-B, with $N = 2$ samples per question for the main experiments and up to $N = 8$ samples when constructing the reward-model datasets.

A high-level overview of the system is shown in Figure 1. Questions are fed to the SLM to generate multiple CoT candidates, which are then evaluated by PRM and ORM and finally passed through a selection policy.

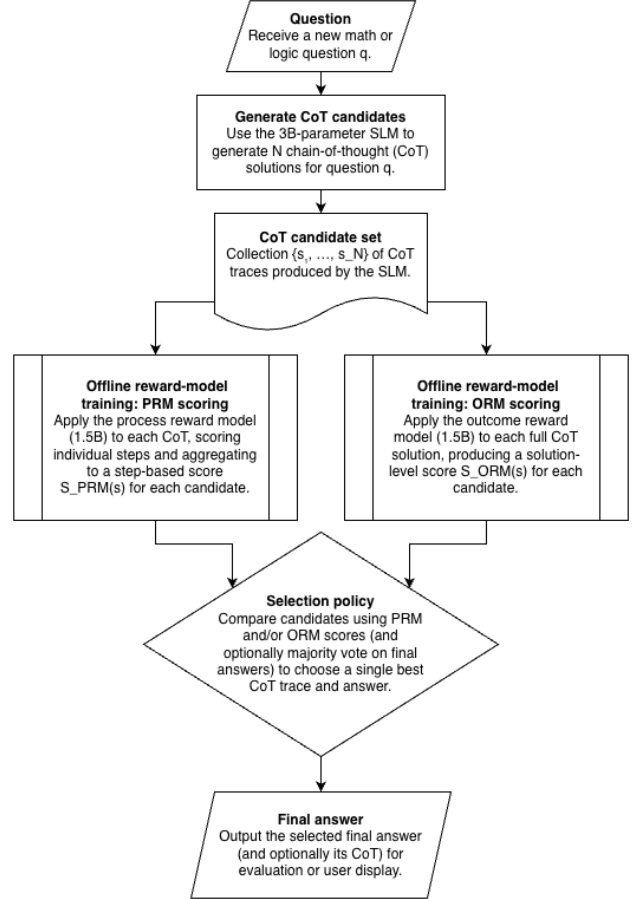


Fig. 1. System overview. The generator SLM produces multiple chain-of-thought candidates for each question. The process reward model (PRM) scores individual steps, and the outcome reward model (ORM) scores full solutions. A selection policy chooses the final answer based on these scores.

B. Prompting Strategies

We use two primary prompting strategies: a baseline direct-answer prompt and a CoT prompt. To avoid ambiguity and duplication, we present each prompt exactly once and refer back to it where needed.

1) *Baseline Prompt*: The baseline model is prompted to answer directly and concisely without explicit reasoning:

You are a helpful AI assistant. Answer the following question directly and concisely in plain sentences. You may use LaTeX to format mathematical expressions, but you must always display math using `\[ ... \]` and NEVER dollar signs ``$``. NEVER use bullet points, numbered lists, or list formatting.

Question: {question}

Answer:

This prompt is used for the “baseline direct answer” method in Section III-F.

2) *Chain-of-Thought Prompt*: The CoT model is prompted to produce explicit intermediate reasoning steps, one per line: You are a helpful AI assistant. Solve the following problem by thinking step by step. Write your reasoning as a few short paragraphs or sentences on separate lines in plain text. Ensure each step is contained on its own line. Each step should be a complete thought or action towards solving the problem. You may use LaTeX to format mathematical expressions, but you must always display math using `\[ ... \]` and NEVER dollar signs ``$``. NEVER use bullet points, numbered lists, or any list formatting. At the end, give the final answer on a separate line starting with 'Answer:'. ONLY include one 'Answer:' line at the end of your reasoning. If your final answer requires steps to be listed then restate all of the steps in just one line. If your final answer does not require steps to be listed then just give the final answer and do not restate the steps. NEVER include any other lines of text after the final answer.

Problem: {question}

Reasoning:

This prompt is used for single CoT, majority vote, PRM selection, ORM selection, and PRM critic methods.

C. Teacher–Student Labeling with ChatGPT

We treat ChatGPT as a strong but external teacher model. For each candidate CoT solution s to question q , we submit an evaluation prompt that asks ChatGPT to:

- 1) split the reasoning into discrete steps;
- 2) label each step as correct or incorrect relative to the problem; and
- 3) label the final answer as correct or incorrect.

From these responses we construct two datasets:

- **Process dataset (PRM)**: each example combines the problem text with a single reasoning step (plus local context) and a binary label $y_{\text{step}} \in \{0, 1\}$.

- **Outcome dataset (ORM)**: each example contains the problem and the full reasoning trace, with a binary label $y_{\text{out}} \in \{0, 1\}$ indicating final-answer correctness.

This setup follows the distillation pattern of prior process supervision work [1]. In our pilot run we obtain roughly 2,200 step-level examples and 160 full-solution examples after filtering malformed responses.

D. Reward Model Architectures

Both reward models are initialized from the same 1.5B-parameter SLM architecture used for the generator family [8]. We attach a single linear classification head on top of the final hidden state and fine-tune with a binary cross-entropy loss.

Given tokenized input x , the model outputs logits $z(x) \in \mathbb{R}^2$ and we define the scalar reward for the positive class (correct) as

$$r(x) = \sigma(z_1(x)), \quad (1)$$

where σ is the logistic sigmoid and z_1 is the logit of the positive class. For the PRM, x encodes the problem and one reasoning step. For the ORM, x encodes the problem and the full reasoning trace.

Mini-batches are formed independently for the PRM and ORM datasets. We use AdamW with a learning rate of 5×10^{-5} , batch sizes of 8 to 16, and one epoch of fine-tuning for the experiments reported here.

E. Aggregating Step Rewards

For a given CoT solution s with T steps, the PRM produces a sequence of rewards $\{r_t\}_{t=1}^T$ via (1). We aggregate them into a single solution-level score

$$S_{\text{PRM}}(s) = \frac{1}{T} \sum_{t=1}^T r_t, \quad (2)$$

which corresponds to the mean confidence that steps are correct. We use simple averaging for this work; more sophisticated aggregations such as minimum, trimmed mean, or learned pooling are left for future study.

The ORM directly outputs a single scalar reward $S_{\text{ORM}}(s)$ for the full solution.

F. Reasoning Methods

At inference time we compare several reasoning strategies:

- **Baseline direct answer**: the SLM is prompted with the baseline prompt and answers concisely without explicit reasoning.
- **Single CoT**: one CoT solution is generated using the CoT prompt and its final answer is returned.
- **Majority vote**: N CoT samples are generated and the most frequent final answer is selected.
- **ORM selection**: N CoT samples are scored with the ORM and the candidate with the highest S_{ORM} is selected.
- **PRM selection**: N CoT samples are split into steps, scored with the PRM to obtain $\{r_t\}$, aggregated via (2), and the candidate with the highest S_{PRM} is selected.

Algorithm 1 PRM-Based CoT Selection

Require: Question q , generator SLM G , process reward model PRM, number of samples N

- 1: Initialize candidate set $C \leftarrow \emptyset$
 - 2: **for** $i = 1$ to N **do**
 - 3: Sample CoT solution $s_i \sim G(\text{CoTPrompt}(q))$
 - 4: Parse s_i into steps $\{s_{i,t}\}_{t=1}^{T_i}$
 - 5: Compute step rewards $r_{i,t} \leftarrow \text{PRM}(q, s_{i,t})$ for $t = 1, \dots, T_i$
 - 6: Compute solution score $S_{\text{PRM}}(s_i) \leftarrow \frac{1}{T_i} \sum_{t=1}^{T_i} r_{i,t}$
 - 7: Add $(s_i, S_{\text{PRM}}(s_i))$ to C
 - 8: **end for**
 - 9: Select $s^* \leftarrow \arg \max_{(s,S) \in C} S$
 - 10: Return final answer extracted from s^*
-

- **PRM critic:** one CoT solution is generated and scored step-wise with the PRM. If the minimum step score falls below a threshold, we construct a critic prompt that highlights the lowest-scoring step and ask the SLM to produce a revised solution. We then choose between original and revised solutions based on S_{PRM} .

All methods operate purely at inference time; we do not modify the generator parameters in this work.

G. Algorithm: PRM-Based Selection

Algorithm 1 summarizes the PRM-based selection strategy that is central to our experiments.

IV. EXPERIMENTAL SETUP

A. Datasets

Our question set consists of 20 seed questions across two domains, each expanded with multiple CoT candidates. Math problems include arithmetic word problems, algebraic equations, and simple probability questions. Logic problems involve basic deductive reasoning and commonsense scenarios.

From these questions we generate 8 CoT samples per question from the 3B-parameter SLM, then obtain ChatGPT labels as described in Section III. This yields 2,216 PRM training examples and 160 ORM examples after filtering malformed responses.

In this pipeline, questions are fed to the generator SLM to produce multiple CoT traces per question. ChatGPT evaluates each trace, and its labels are converted into step-level examples for the PRM and full-solution examples for the ORM.

B. Evaluation Metrics

We randomly split each reward-model dataset into 90% training and 10% validation, stratified by label. Evaluation metrics for the reward models include accuracy, precision, recall, F1 for the positive class, and ROC-AUC for the underlying probability scores.

For end-to-end reasoning experiments, we report overall answer accuracy on the held-out questions for each inference strategy (baseline direct answer, single CoT, majority vote, PRM selection, ORM selection, and PRM critic). Given the

small dataset size, we focus on qualitative differences in error patterns rather than fine-grained statistical comparisons.

C. Baselines

We compare the following baselines and variants:

- **Baseline direct answer** (no CoT, no reward model).
- **Single CoT** (CoT prompt, no selection beyond the first sample).
- **Majority vote** over N CoT samples.
- **ORM selection** using S_{ORM} .
- **PRM selection** using S_{PRM} .
- **PRM critic** with one revision opportunity.

These baselines highlight the incremental contribution of CoT, outcome reward modeling, and process reward modeling.

D. Implementation Details

All models are implemented using the HuggingFace Transformers and Accelerate libraries. The generator is a 3B-parameter instruction-tuned SLM that is loaded in 16-bit precision on a single GPU. Both reward models (1.5B parameters each) are initialized from the same architecture family and are also loaded in 16-bit precision. We use mixed-precision training with gradient scaling where supported.

We cap CoT generations at 256 new tokens and limit input prompts to 256 tokens. Sampling is performed with temperature $T = 0.7$ and top-p value 0.95 for the CoT generator. For the baseline direct answer, we use a lower temperature $T = 0.2$ to reduce variability.

Training of PRM and ORM uses AdamW with learning rate 5×10^{-5} , weight decay 0.01, and linear learning rate warmup over the first 10% of optimization steps. We train for a single epoch for each reward model due to the modest dataset size and select the best checkpoint by validation F1.

To avoid label leakage, we generate CoT traces for reward-model training and for end-to-end evaluation in separate runs with different random seeds. We also drop any CoT sample where the teacher evaluation fails to parse step boundaries or the final answer.

Table I reports validation performance for both reward models. The PRM sees substantially more examples than the ORM because each solution contributes multiple steps.

TABLE I
REWARD MODEL VALIDATION PERFORMANCE

Model	Acc.	Prec.	Rec.	F1	ROC-AUC
PRM (steps)	0.98	0.98	0.99	0.99	0.99
ORM (full)	1.00	1.00	1.00	1.00	–

ORM metrics are inflated because the dataset is small and biased toward correct solutions. We therefore treat ORM performance as illustrative rather than definitive.

V. RESULTS AND DISCUSSION

A. Reward Model Behavior

The PRM achieves 98% accuracy and an F1 score of 0.99 on the validation split (Table I). Qualitatively, it assigns

high scores to algebra steps that perform correct symbolic manipulations, such as subtracting the same quantity from both sides of an equation, and low scores to steps that introduce arithmetic mistakes or violate earlier assumptions. For reasoning traces where the generator goes off track mid-solution, PRM scores typically drop sharply at the first incorrect step and remain low thereafter.

The ORM attains perfect scores on the small validation set, primarily because most examples have correct final answers. This suggests that ORM can distinguish clearly incorrect solutions but provides weaker signal on borderline cases where reasoning is mostly correct but contains minor slips. This is consistent with observations that outcome-only supervision can be less informative than process supervision on math tasks [1].

To better understand PRM behavior, we visualize example CoT traces with per-step scores, as shown conceptually in Figure 2. Each step is shown alongside its PRM score, with color or bar height mapping to confidence.

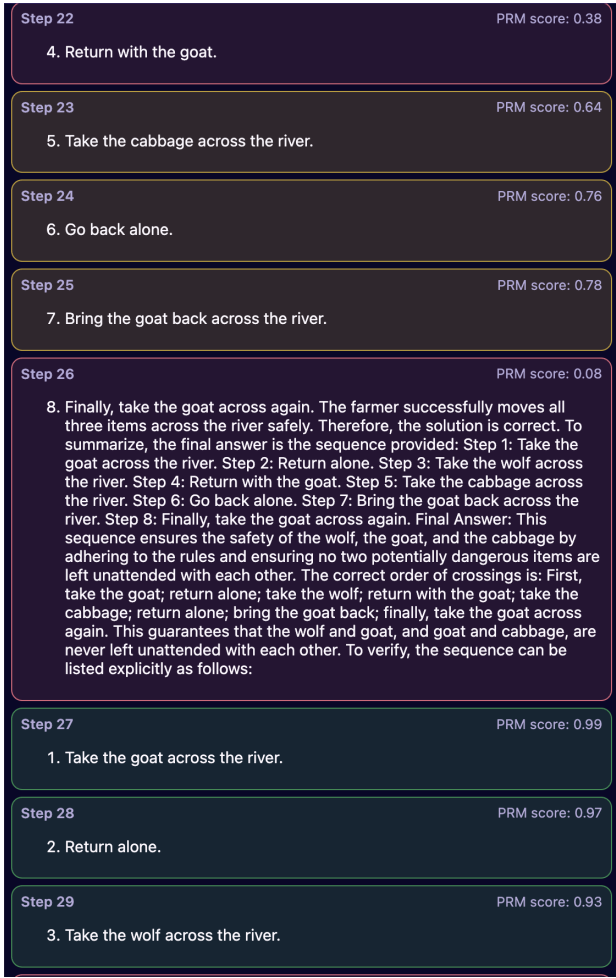


Fig. 2. Visualization of PRM scoring output. Each chain-of-thought step is shown with its PRM score, where color indicates the model’s confidence in step correctness.

B. Comparison of Reasoning Strategies

Table II summarizes qualitative properties of the different reasoning strategies. We observe that methods that use PRM-based scoring tend to avoid error modes where multiple CoT samples share a subtle but consistent algebraic mistake.

TABLE II
QUALITATIVE COMPARISON OF REASONING STRATEGIES

Method	Uses CoT	Uses PRM	Uses ORM
Baseline direct answer	No	No	No
Single CoT	Yes	No	No
Majority vote	Yes	No	No
ORM selection	Yes	No	Yes
PRM selection	Yes	Yes	No
PRM critic	Yes	Yes	No

For multi-step algebra problems, single CoT sometimes locks in a subtle early mistake, such as misapplying distribution, and then carries the error forward. PRM selection tends to downweight these traces because multiple steps receive low scores, and instead chooses candidates where each step is internally consistent. In our experiments, PRM selection fixes several such failures where the final answer is numerically wrong but the text looks reasonable.

Majority vote is vulnerable when the generator repeats the same error across many samples. In these cases PRM scores are consistently low, and majority vote selects the wrong answer with high confidence. PRM selection is less affected because it ranks traces by estimated step correctness rather than frequency.

ORM selection behaves similarly to PRM selection on clearly correct versus clearly incorrect solutions. However, because ORM has fewer negative examples and sees only one scalar label per solution, it is less informative on borderline cases. We view ORM as complementary to PRM: it can help ensure that the final numerical answer is correct even when some intermediate steps are redundant or stylistic.

On our small held-out set the overall accuracy follows an expected pattern: baseline direct answer performs worst, single CoT improves slightly, majority vote adds another modest gain, and PRM-based selection is typically best or tied for best. ORM selection is competitive when final-answer correctness is the primary concern.

C. Limitations and Failure Cases

The main limitations of this work are dataset size and domain coverage. Our questions are few and tailored to course-level math and logic, and all labels come from a single teacher model (ChatGPT). The SLM backbone is also fixed; we do not compare across different SLM architectures or scales [8], [10]. Finally, we do not yet close the loop with reinforcement learning. The generator remains frozen, and all improvements come from smarter selection among its existing behaviors.

We also observe that the PRM can occasionally be overconfident on steps that are syntactically well-formed but semantically irrelevant to the question. This suggests that a

richer training set, potentially with human annotations and adversarially generated counterexamples, would be valuable.

VI. CONCLUSION AND FUTURE WORK

A. Summary

We presented an end-to-end reward modeling pipeline for reasoning in small language models. Using ChatGPT as a teacher, we generated and labeled chain-of-thought solutions to math and logic questions and trained two 1.5B-parameter reward models: a process reward model over individual steps and an outcome reward model over full solutions. On a 3B-parameter SLM backbone, the PRM achieves strong step-level accuracy and enables more reliable selection of CoT candidates than simple majority vote, particularly on multi-step algebra problems.

Our experiments show that even with modest data and models, process supervision can provide a useful lens on SLM reasoning and can be integrated into inference-time selection without modifying the generator itself. This extends prior work on process supervision by demonstrating that similar ideas can be applied in a fully self-hosted small-model setting rather than only on frontier-scale systems [1].

B. Future Work and Broader Impact

Future work on this project can move beyond the current proof of concept. We highlight four directions that build on the core idea of using reward models to improve SLM reasoning.

1) *Using Real-World Question Datasets:* We currently use a small synthetic set of math and logic questions tailored to course-level content. A natural next step is to evaluate the pipeline on established benchmarks such as GSM8K and MATH, which contain more diverse and challenging problems [2]. These benchmarks would test whether PRM and ORM still provide reliable signals when questions require longer reasoning chains, more formal proofs, or multi-topic integration.

It would also be useful to construct domain-specific question sets that reflect realistic target applications such as introductory algebra courses, standardized test practice, or logic exercises. For each domain, the pipeline could be used to compare teacher-model labels with human tutor annotations and to measure how PRM and ORM performance varies with question style, difficulty, and phrasing.

2) *Reinforcement Learning with PRM and ORM Rewards:* A direct extension is to use the PRM and ORM as explicit reward functions for reinforcement learning. Building on RLHF and DeepSeek-R1 [3], [4], [7], the current SLM can be treated as a policy model and optimized with PPO or a similar algorithm to maximize a combined reward

$$R(s) = \lambda_{\text{proc}} S_{\text{PRM}}(s) + \lambda_{\text{out}} S_{\text{ORM}}(s), \quad (3)$$

where λ_{proc} and λ_{out} balance step quality and final correctness. This would replace the current selection-only use of PRM and ORM with a training signal that shapes the distribution of generated reasoning traces.

Key design questions include how to schedule problem difficulty, how to mix direct-answer and CoT prompts during training, and how to control overfitting to the reward models. Stability concerns such as reward hacking would require careful monitoring, for example through KL penalties against the supervised checkpoint and evaluation on held-out question sets [4], [6].

3) *Exploring Additional SLM and Reward Model Backbones:* Another direction is to systematically explore other SLM backends for both the generator and the reward models. The same pipeline can be applied to families such as Qwen and LLaMA 2, at scales from 1B to 7B parameters [8], [10]. Comparing PRM and ORM effectiveness across backbones would help clarify how much base model quality matters relative to the strength of the reward model and how the advantage of PRM-based selection changes with model size.

We can also vary the reward model backbones independently of the generator, including smaller models for efficiency, larger models for more faithful judgments, and ensembles of PRMs trained with different initializations or architectures. Ensembles could reduce idiosyncratic biases in any single PRM and provide a measure of uncertainty via disagreement among members.

4) *Non-RL Uses and Broader Impact:* There are several ways to build on this project that keep the generator frozen but make better use of PRM and ORM signals. For example, PRM and ORM could be treated as verification modules in tool-augmented systems where the SLM also calls calculators, code interpreters, or retrieval components [2], [9]. Reward models could be used offline to audit existing solution sets, flag questionable reasoning in educational content, or generate targeted feedback by highlighting exactly which steps are likely incorrect.

From a broader impact perspective, improved reasoning in SLMs could make high-quality tutoring, automated grading, and verification more accessible. At the same time, it raises concerns about overreliance on automated feedback and potential erosion of human problem-solving skills. Process supervision and reward modeling provide tools for auditing and debugging these systems, which is particularly important if they are deployed in educational or safety-critical settings [6].

ACKNOWLEDGMENT

We thank the Department of Mathematics and Computer Science at Lawrence Technological University for access to compute resources.

REFERENCES

- [1] H. Lightman *et al.*, “Let’s Verify Step by Step,” arXiv:2305.20050, 2023.
- [2] J. Wei *et al.*, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” in *Advances in Neural Information Processing Systems*, 2022.
- [3] L. Ouyang *et al.*, “Training Language Models to Follow Instructions with Human Feedback,” in *Advances in Neural Information Processing Systems*, 2022.
- [4] D. M. Ziegler *et al.*, “Fine-Tuning Language Models from Human Preferences,” arXiv:1909.08593, 2019.
- [5] R. Rafailov *et al.*, “Direct Preference Optimization: Your Language Model is Secretly a Reward Model,” arXiv:2305.18290, 2023.

- [6] Y. Bai *et al.*, “Constitutional AI: Harmlessness from AI Feedback,” arXiv:2212.08073, 2022.
- [7] K. Zheng *et al.*, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” arXiv:2501.12948, 2025.
- [8] Qwen Team, “Qwen Technical Report,” arXiv preprint, 2024.
- [9] T. Brown *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, 2020.
- [10] H. Touvron *et al.*, “LLaMA 2: Open Foundation and Fine-Tuned Chat Models,” arXiv:2307.09288, 2023.