

What Are Step-Level Reward Models Rewarding? Counterintuitive Findings from MCTS-boosted Mathematical Reasoning

Yiran Ma^{1*}, Zui Chen^{2*}, Tianqiao Liu³, Mi Tian³, Zhuo Liu⁴, Zitao Liu^{5†}, Weiqi Luo⁵

¹Zhejiang University, Hangzhou, China

²ShanghaiTech University, Shanghai, China

³TAL Education Group, Beijing, China

⁴University of Rochester, New York, USA

⁵Jinan University, Guangzhou, China

mayiran@zju.edu.cn, chenzui@shanghaitech.edu.cn, {liutianqiao1, tianmi}@tal.com, zhuo.liu@rochester.edu, {liuzitao, lwq}@jnu.edu.cn

Abstract

Step-level reward models (SRMs) can significantly enhance mathematical reasoning performance through process supervision or step-level preference alignment based on reinforcement learning. The performance of SRMs is pivotal, as they serve as critical guidelines, ensuring that each step in the reasoning process is aligned with desired outcomes. Recently, AlphaZero-like methods, where Monte Carlo Tree Search (MCTS) is employed for automatic step-level preference annotation, have proven particularly effective. However, the precise mechanisms behind the success of SRMs remain largely unexplored. To address this gap, this study delves into the counterintuitive aspects of SRMs, particularly focusing on MCTS-based approaches. Our findings reveal that the removal of natural language descriptions of thought processes has minimal impact on the efficacy of SRMs. Furthermore, we demonstrate that SRMs are adept at assessing the complex logical coherence present in mathematical language while having difficulty in natural language. These insights provide a nuanced understanding of the core elements that drive effective step-level reward modeling in mathematical reasoning. By shedding light on these mechanisms, this study offers valuable guidance for developing more efficient and streamlined SRMs, which can be achieved by focusing on the crucial parts of mathematical reasoning.

Introduction

Large Language Models (LLMs) have demonstrated their remarkable capabilities across a wide range of tasks, such as information extraction, natural language understanding, etc (Zhao et al. 2023), totally revolutionizing the deep learning community. Among these capabilities, reasoning stands out as a critical area of focus, especially mathematical reasoning, which needs to be further improved due to its complex nature. Numerous studies have shown that multi-step reasoning often facilitated through Chain-of-Thought (CoT)

*These authors contributed equally. Work was done during their internships at TAL Education Group.

†Zitao Liu is the corresponding author.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

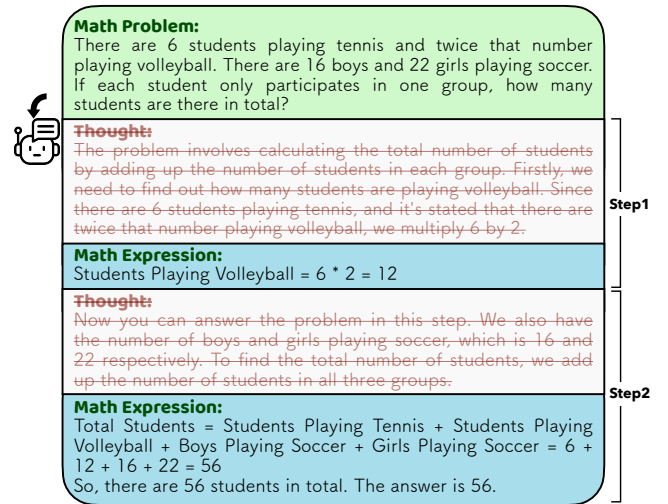


Figure 1: Each step in an LLM’s process of solving mathematical problems can be divided into the thought process and the execution of corresponding calculations. We find that natural language descriptions of the thought processes are not essential for step-level reward modeling.

prompting, can significantly enhance model performance on reasoning tasks (Zhou et al. 2023; Besta et al. 2024; Ding et al. 2023; Yao et al. 2024; Wang et al. 2022; Wei et al. 2022; Zheng et al. 2024; Li et al. 2024; Zhan et al. 2024).

Recently, guided tree-search methods further improved reasoning performance by exploring various reasoning paths through online simulation to identify the optimal solution paths (Hao et al. 2023, 2024; Feng et al. 2023). Although a better reasoning path leads to a better performance, the length of these reasoning chains leads to an exponential increase in the search space, resulting in substantial computational costs. Given the high expense of LLM inference, performing an online tree search for each reasoning problem introduces repeated and unnecessary overhead.

To address this issue, step-level reward models (SRM)

was proposed to improve search efficiency. Lightman et al. (2023) introduced the process reward model (PRM), which employs human-annotated step-level scores for reward modeling, and Ma et al. (2023) further demonstrated the effectiveness of SRMs in math reasoning and coding tasks. Then, Math-Shepherd (Wang et al. 2024), systematically generates step-level preference data through exhaustive reasoning process traversal to train reward models and reinforce the model’s capabilities. More recently, inspired by AlphaZero, Monte Carlo Tree Search (MCTS) (Xie et al. 2024; Chen et al. 2024a,b) was then used for collecting preferences more efficiently because of its capability of balancing exploration and exploitation. These trained SRMs can effectively enhance reasoning performance by either assisting step-level preference alignment with proximal policy optimization (PPO) during training stage or serving as step verifiers during inference stage.

Despite the significant achievements in mathematical reasoning performance achieved by the SRMs constructed by MCTS-based method, the exact workings of these reward models and what they are truly rewarding remain unclear. Brain and cognitive scientists have argued that diverse thinking and reasoning processes do not necessarily rely on natural language. (Fedorenko, Piantadosi, and Gibson 2024). A skilled human mathematician, for instance, can determine whether a mathematical expression is logically coherent and numerically correct without the participation of the natural language. Building on this idea, our research explores a similar hypothesis for LLMs: that **natural language descriptions of thought processes are not essential for mathematical reasoning within these models**. We suppose that LLMs can be trained to recognize preferences for mathematical language directly during problem-solving, without relying on natural language descriptions. This implies that LLMs might be capable of understanding and processing mathematical reasoning through the intrinsic structure of mathematical language, potentially leading to more efficient and focused training methods that bypass the need for natural language explanations. Furthermore, it is believed that incorrect solutions often arise from wrong mathematical calculations or logical errors (Zhang et al. 2024), with the latter being more challenging (Chen et al. 2024a). Therefore, we further investigate the effectiveness of SRMs in evaluating logical coherence in pure mathematical language, demonstrating that the improvements are not merely the result of encouraging correct calculations within a single step. Additionally, and somewhat surprisingly, we found that SRMs struggle to learn how to evaluate logical coherence in natural language. This will further support that natural language is not necessary for step-level reward modeling.

To investigate the respective roles of natural language and mathematical language in step-level reward modeling, we decompose each step of the reasoning path into two components: natural language descriptions of thought processes and math expressions (Figure 1). The ablation studies are conducted by selectively removing different parts from the inputs of the SRMs. This decomposition mirrors the human problem-solving process in mathematics, which typically involves an initial phase of thinking through the problem, fol-

lowed by the execution of calculations based on that thought process. The thought processes include the strategy to be taken in that step, while the calculations are the executions of the thought processes. In other words, our decomposition aims to separate the natural language (composing the ‘thoughts’) from the mathematical expressions (contained in the execution of ‘thoughts’). This framework aims to foster a deeper understanding of the role of natural language for step-level reward modeling.

To summarize, our experiments support that SRMs appear to have some intrinsic affinity for mathematical expression, not natural language. Specifically, we propose the following key insights.

1. Natural language descriptions of thought processes are not necessary for successful step-level reward modeling.
2. SRMs not only promote accurate calculations within individual steps but also effectively assess the challenging logical coherence in mathematical language.
3. Assessing logical coherence in natural language is difficult, and SRMs often struggle with this task.

Preliminaries

Markov Decision Process

Definition A Markov Decision Process (MDP) is a mathematical framework used to model decision-making problems. This framework is fundamental for addressing a wide range of reinforcement learning (RL) problems where the outcomes are partially random and partially controllable. An MDP is defined by a tuple (S, A, P, R, γ) , where:

- S is the set of states.
- A is the set of actions.
- P is the transition probability function, $P(s_{t+1}|s_t, a_t)$, which defines the probability of transitioning to state s_{t+1} given the current state s_t and action a_t .
- R is the reward function, $R(s_t, a_t, s_{t+1})$, which defines the reward received after transitioning from state s_t to state s_{t+1} by taking action a_t .
- γ is the discount factor, which determines the importance of future rewards.

Bellman Expectation Equation For **state value function** $V(s)$, the Bellman Expectation Equation is:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [\mathbb{E}_{s' \sim P(\cdot|s,a)} [R(s, a, s') + V^\pi(s')]]$$

For **state-action value function** $Q(s, a)$, the Bellman Expectation is:

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} [R(s, a, s') + \mathbb{E}_{a' \sim \pi(\cdot|s')} [Q^\pi(s', a')]]$$

Optimal Value Functions The optimal value functions are defined as:

$$\begin{aligned} V^*(s) &= \max_{\pi} V_{\pi}(s) \\ Q^*(s, a) &= \max_{\pi} Q_{\pi}(s, a) \end{aligned} \quad (1)$$

Therefore, the relationship between the optimal value functions and the Bellman Optimality Equation is:

$$V^*(s) = \max_a Q^*(s, a) \quad (2)$$

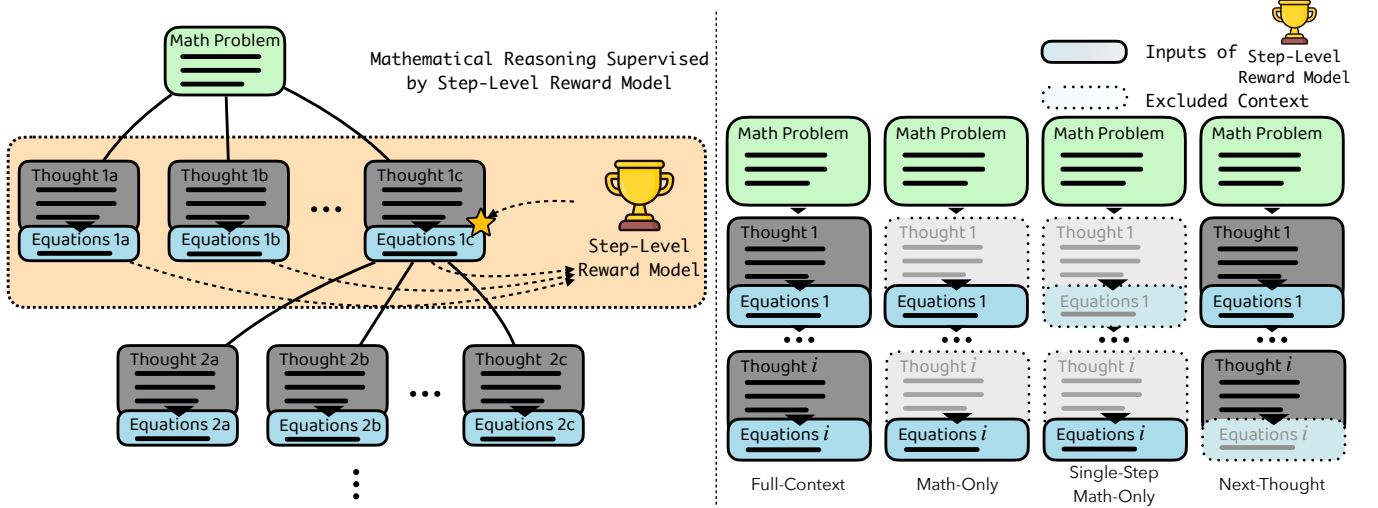


Figure 2: Illustration of the role of SRMs in mathematical reasoning and the SRMs with different input structures we investigate.

Setup

LLM’s Math Reasoning as MDP: Our Definition

Figure 2 shows the mathematical reasoning process with each step decomposed into thought and math expressions. Specifically, our MDP definition is as follows:

$$\text{MDP} = (S, A, P, R)$$

where:

- **State** The state space S consists of states defined as $s_i = (T_k, E_k)_{k=0}^i$, representing a sequence of thoughts T_k and equations E_k up to step i .
- **Action** The action space A consists of actions defined as $a_i = T_{i+1}$, representing the natural language descriptions of the subsequent thought proposed by the LLM.
- **State Transition** $P(s_{i+1}|s_i, a_i)$ is the state transition function, defining the probability of transitioning to state s_{i+1} from state s_i after taking action a_i . This function is implemented by the LLM generating the corresponding math expression E_{i+1} based on the next thought $a_i = T_{i+1}$ and the current state $s_i = (T_k, E_k)_{k=0}^i$.
- **Reward Function** $R(s_i, a_i, s_{i+1})$ is the reward function, defining the immediate reward received after transitioning to state $s_{i+1} = (T_k, E_k)_{k=0}^{i+1}$ from state s_i by taking action a_i . We define the reward up to state s_{i+1} based on whether it can lead to the correct final answer:

$$R(s_i, a_i, s_{i+1}) = \begin{cases} 1, & \text{final answer is correct} \\ 0, & \text{final answer is incorrect} \end{cases} \quad (3)$$

Additionally, policy $\pi(a_i|s_i)$ is implemented by the LLM generating the thought of the next step $a_i = T_{i+1}$ based on the current state $s_i = (T_k, E_k)_{k=0}^i$. According to Equation (1), the goal of an agent is to maximize $V_\pi(s_i)$ or $Q_\pi(s_i, a)$ by generating the correct thoughts T in each step.

In summary, a language model plays a dual role in the MDP framework:

1. **As an Agent** The LLM is responsible for making decisions by selecting appropriate actions (next thoughts T_{i+1}) at each state, following the policy $\pi(a_i|s_i)$.
2. **As a World Model** The LLM also acts as the world model $P(s_{i+1}|s_i, a_i)$ by predicting action outcomes (state transitions) using its internal knowledge and training data. It simulates the environment of mathematical reasoning by executing thought T_{i+1} through corresponding calculations, thus providing the prediction of new states s_{i+1} .

MCTS for Step-Level Preference Collection

Understanding the natural correspondence between math reasoning and MDP, we can readily use MCTS for efficient step-level preference collection. The MCTS starts from a root node s_0 , which is a math problem in mathematical reasoning tasks. Then, each new node corresponds to a state update. Each iteration of MCTS can be divided into four phases: Selection, Expansion, Rollout, and Back-propagation.

1. **Selection.** The selection phase in MCTS involves traversing the tree from the root node s_0 (the initial math problem) to a leaf node using a selection policy. This policy, typically the Upper Confidence Bound for Trees (UCT) formula, balances exploration and exploitation. At node s_i , the next node is chosen by:

$$s_{i+1}^* = \arg \max_{s_{i+1}} \left[\frac{c(s_{i+1})}{N(s_{i+1})} + w_{\text{exp}} \cdot \sqrt{\frac{\log N(s_i)}{N(s_{i+1})}} \right], \quad (4)$$

where $c(s_{i+1})$ is the correct counts, $N(s_i)$ and $N(s_{i+1})$ are visit counts, and w_{exp} balances exploration and exploitation. This process continues until an unexplored node is found.

2. **Expansion.** Upon reaching a leaf node, n new candidate actions (thoughts) $\{a_i^j \mid j = 1, \dots, n\}$ are generated by

the agent given the current state s_i . Given the candidate actions (thoughts), the world model will execute them through mathematical calculations, constructing the new candidate states $\{s_i^j \mid j = 1, \dots, n\}$. These candidate states are added as child nodes to the current node to expand the tree, allowing for a broader exploration of potential problem-solving paths.

3. **Rollout.** The rollout phase simulates the reasoning process from the newly expanded node to a terminal state or predefined maximum depth. The score of a node is then obtained according to Equation (3). This procedure estimates the scores of the new nodes according to the simulation results, informing the back-propagation phase.
4. **Back-propagation.** Results from the rollout are propagated back up the tree to update values and visit counts of each node. Starting from the final state, the effectiveness of the problem-solving process updates the value $V(s)$ of each state. This procedure improves the selection policy for future iterations.

After completing MCTS, step-level preference pairs can be gathered by comparing the values of the nodes in each tree.

Step-level Reward Modeling

After collecting all the preference pairs, step-level reward models can be constructed through contrastive learning. Based on our MDP definition, an SRM is regarded as the action-value function $Q(s, a)$ or the value function $V(s)$. Specifically, we investigate different reward models for ablation studies, where reward models take different inputs to evaluate the ongoing reasoning process. Accordingly, we define four reward models (Figure 2-right) for the ablation study:

- **Full-Context Step-level Reward Model (FC-SRM)**
This model takes both the thoughts and math expressions of the current state as input.

$$V_1(s_i) = V_1((T_k, E_k)_{k=0}^i) \quad (5)$$

- **Math-Only Step-level Reward Model (MO-SRM)**
This model takes only the math expressions of the current state as input, excluding the natural language descriptions of thought processes.

$$V_2(s_i) = V_2((E_k)_{k=0}^i) \quad (6)$$

- **Single-Step Math-Only Step-level Reward Model (SSMO-SRM)** This model takes only the newest math expression of the ongoing reasoning process as input, excluding the natural language and all the previous math expressions.

$$V_3(s_i) = V_3(E_i) \quad (7)$$

- **Next-Thought Step-level Reward Model (NT-SRM)**
This model takes both the thoughts and math expressions of the current state as input, and evaluates the next thought. According to our definition, the next thought is the action taken by the agent. Thus this reward model is the action-value function under our MDP definition of mathematical reasoning.

$$Q(s_i, a_i) = Q((T_k, E_k)_{k=0}^i, T_{i+1}) \quad (8)$$

Beam Search with Step-Level Reward Model

Given the SRMs trained on the preference data, it is commonly used for step-level preference alignment to update the policy. The purpose of this procedure is to generate the best action through the updated policy π' , thereby reducing the overhead caused by online MCTS. It is also possible to update the world model P with these preference pairs as better accuracy indicates better mathematical performance.

Algorithm 1: Beam Search Algorithm

Require: Initial state s_0 , beam size B , candidate count c

- 1: Initialize beam $\mathcal{B} \leftarrow \{s_0\}$
- 2: **while** \mathcal{B} is not empty **do**
- 3: Initialize empty list $\mathcal{B}_{\text{next}} \leftarrow \emptyset$
- 4: **for** each state s_i in \mathcal{B} **do**
- 5: Generate a set of candidate actions $\{a_i^1, a_i^2, \dots, a_i^c\}$ based on s_i
- 6: **for** each action a_i^j in $\{a_i^1, a_i^2, \dots, a_i^c\}$ **do**
- 7: Compute the next state $s_{i+1}^j \leftarrow P(s_{i+1} \mid s_i, a_i^j)$
- 8: Evaluate the score of s_{i+1}^j
- 9: Add s_{i+1}^j to $\mathcal{B}_{\text{next}}$
- 10: **end for**
- 11: **end for**
- 12: Sort $\mathcal{B}_{\text{next}}$ by score and keep the top B states
- 13: Update beam $\mathcal{B} \leftarrow$ top B states from $\mathcal{B}_{\text{next}}$
- 14: **end while**
- 15: **return** the best state from the final beam

As this study focuses on the SRMs, our experiments will not include the preference alignment procedure. Instead, we can use the SRMs as the scoring function during beam search (BS) Algorithm 1 for simplification. This simplification excludes potential uncertainties in the alignment process, providing a more straightforward understanding of SRMs' effectiveness. **Notably, setting $B = 1$ makes BS effectively become greedy search (GS).**

The greedy search can be regarded as a reasoning process supervised by an SRM (Figure 2-left). Indeed, with an infinite number of samples, the optimal actions and states identified through the policy π and the world model P will converge to the optimal actions and states similar to those generated by the optimal policy π^* in Equation (1), respectively.

$$\lim_{n \rightarrow \infty} P(\arg \max_{\{a_t\}_{t=0}^n Q(s, a_t) = \arg \max_{a \in A_\pi(s)} Q(s, a)) = 1 \quad (9)$$

where $a_t \sim \pi(a \mid s)$ and $A_\pi(s)$ denotes the state space of actions generated by the policy π given state s . Similarly, for states, we also have

$$\lim_{n \rightarrow \infty} P(\arg \max_{\{s'_t\}_{t=0}^n V(s'_t) = \arg \max_{s' \in S(s, a)} V(s')) = 1 \quad (10)$$

where $s_t \sim \mathbb{E}_{a_{t-1} \in \pi(a \mid s_{t-1})} P(s \mid s_{t-1}, a_{t-1})$.

Experiments

Implementation Details

Datasets To construct step-level preference pairs through MCTS, we use the math problems and their corresponding

Agent & World Model	Historical Thoughts	Historical Equations	Next Thoughts	Next Equations	Accuracy (Gain) %	
					GSM8K	MATH
Llama-3-8B-Instruct						
Pass@1 (3-shots)					78.47 (+0.00)	31.16 (+0.00)
+GS w/ SRM (DeepSeek-Math-7B-Base)						
Full-Context SRM	✓	✓	✓	✓	86.20 (+7.73)	38.58 (+7.42)
Math-Only SRM	✗	✓	✗	✓	85.82 (+7.35)	39.64 (+8.48)
Single-Step Math-Only SRM	✗	✗	✗	✓	82.11 (+3.64)	37.46 (+6.30)
Next-Though SRM	✓	✓	✓	✗	79.38 (+0.91)	30.98 (-0.18)
+GS w/ SRM (Qwen2-7B)						
Full-Context SRM	✓	✓	✓	✓	82.94 (+4.47)	35.58 (+4.42)
Math-Only SRM	✗	✓	✗	✓	83.78 (+5.31)	35.10 (+3.94)
Single-Step Math-Only SRM	✗	✗	✗	✓	81.65 (+3.18)	33.08 (+1.92)
Next-Though SRM	✓	✓	✓	✗	81.73 (+3.26)	31.40 (+0.24)

Table 1: SRMs act as step-level scoring functions during GS. Sample $c = 5$ candidates of the subsequent step at each node and use beam size $B = 1$ (greedy search). The agent and the environment model is Llama-3-8B-Instruct. The reward models are trained based on Deepseek-Math-7B-Base or Qwen2-7B.

final answers from the training data of GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021). The accuracies are evaluated on the test data.

Models The reasoning process is conducted by the dialogue between two LLMs. We use the Llama-3-8B-Instruct (Dubey et al. 2024) as both the agent and world model in MCTS because of its excellent ability to follow instructions.

Prompt One LLM (as agent) is instructed to generate natural language descriptions of thoughts, and the other (as world model) is instructed to execute the thoughts. For specific prompts, see Appendix.

Baseline We use Llama-3-8B-Instruct construct the ‘Pass@1’ baseline based on our prompt with 3 shots.

MCTS for Step-Level Preference Collection The MCTS requires the agent sampling $n = 6$ candidate actions at each expansion phase and iterates 500 times on each problem to evaluate the quality of each node. Notably, to avoid the influence of the variation of answer format, we use a supervised fine-tuned (SFT) model based on DeepSeek-Math-7B-Base to assert the correctness of the solution after each rollout during the search. This model is also used in our evaluation pipeline. To strengthen the preferences, only the preference pairs whose difference of value is greater than 0.7 are assumed valid. For detailed hyperparameters, see Appendix.

Reward Training DeepSeek-Math-7B-Base (Shao et al. 2024) or Qwen2-7B (Yang et al. 2024) is used as the base model for SRM training. Each SRM is trained on two instances, with each instance equipped with 8 A800 GPUs. For detailed hyperparameters, see Appendix.

Main Results

After collecting all the step-level preference pairs through MCTS, datasets are constructed for FC-SRM, MO-SRM, SSMO-SRM, and NT-SRM training by selecting the corresponding components in each piece of data. The training

curves are shown in Figure 3. These SRMs are subsequently used as scoring functions in greedy search, the accuracy and absolute gains over baseline are reported in Table 1. The analyses will be included in the following sections.

Do we really need natural language?

Intuitively, one might expect that natural language descriptions provide essential contextual information and aid SRMs’ cognitive understanding. The SRMs with different input formats: full-context (FC) and math-only (MO) are trained to investigate this aspect.

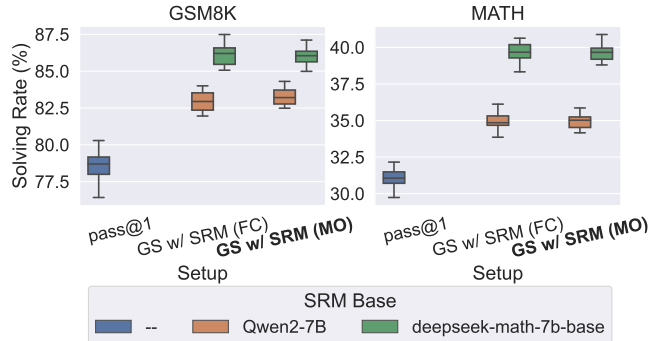


Figure 4: SRMs take only mathematical expressions as input demonstrate the same ability during the greedy search as those take full context as input. The boxplot is obtained through 20 runs over the dataset.

Removing natural language has a minimal effect on step-level reward modeling. FC-SRMs and MO-SRMs exhibit very similar performance in both preference prediction accuracy and greedy search, suggesting that successful step-level reward modeling is not contingent upon natural language descriptions, which is contrary to intuition. Even without the natural language descriptions of thoughts at each step, the

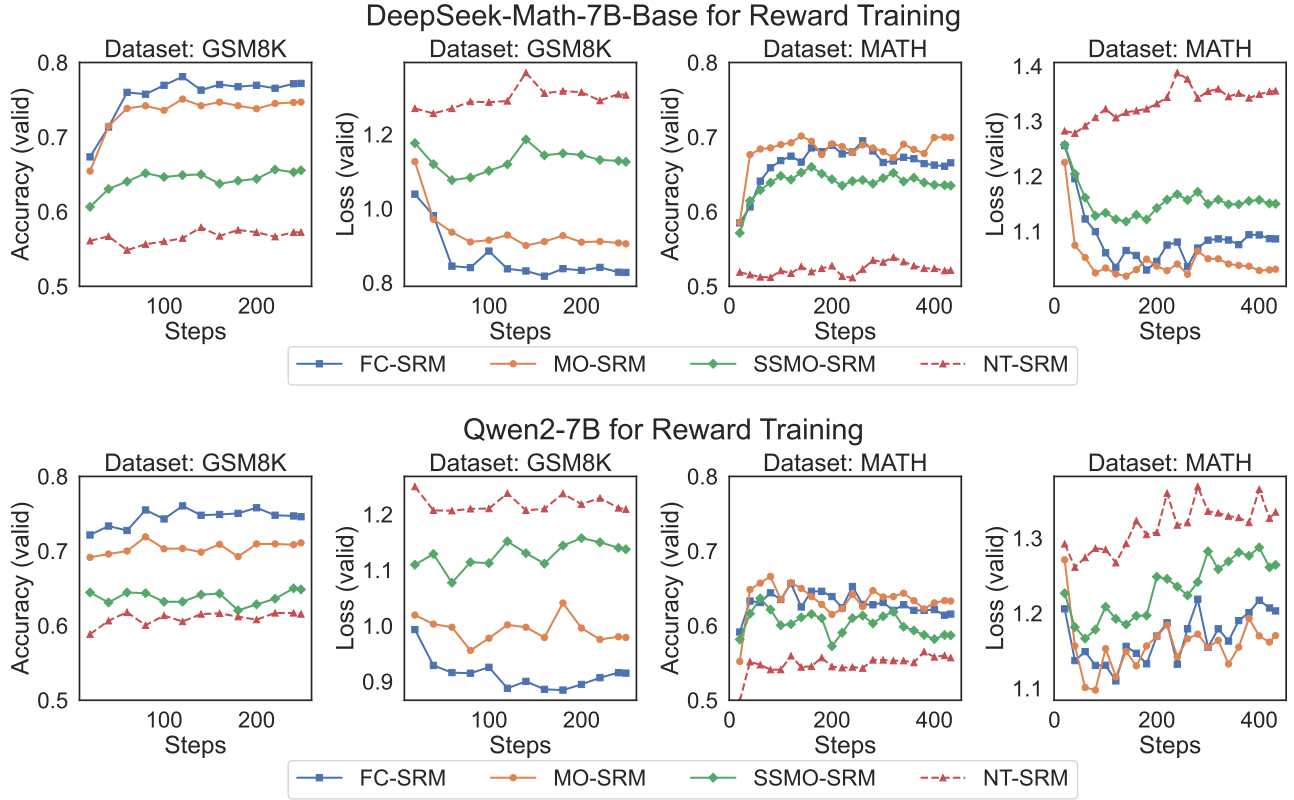


Figure 3: Effect of natural language descriptions and math expressions on step-level reward modeling. The agent and the environment model is Llama-3-8B-Instruct. The reward models are trained based on Qwen2-7B or Deepseek-Math-7B-Base. (Note that the ‘accuracy’ here is the accuracy of preference during reward training.)

MO-SRMs can still be successfully trained (Figure 3). Table 1 and Figure 4 further show the performance of these SRMs when used as scoring functions during greedy search. In setups such as MATH with DeepSeek-Math-7B-Base as the base model of SRM, the MO-SRM (39.64%) can even outperform the FC-SRM (38.58%). We further conducted t-tests to provide a more detailed statistical comparison between the FC-SRMs and MO-SRMs across different datasets and base models. For the GSM8K dataset, the t-test results are $t = -0.18$, $p = 0.86$ for Qwen2-7B, and $t = -0.14$, $p = 0.89$ for deepseek-math-7b-base. For the MATH dataset, the results are $t = 0.79$, $p = 0.44$ for Qwen2-7B, and $t = 0.77$, $p = 0.45$ for deepseek-math-7b-base. In all cases, the p-values are greater than 0.05, indicating that the differences in performance between the FC-SRM and MO-SRM are not statistically significant. These results support the conclusion that omitting natural language from the inputs of SRMs has negligible effects on the effectiveness of SRMs.

Can SRMs evaluate logical coherence in math language?

The success of MCTS-based methods is attributed to the ability to avoid logical and numerical errors. It is commonly believed that logical errors are more difficult to evaluate,

while MCTS-based methods are believed a competitive solution to this challenge by collecting such preferences. In this section, we investigate the role of natural language and mathematical language in assessing the logical coherence included in pure mathematical language by comparing SSMO-SRM, MO-SRM, and NT-SRM.

Specifically, if the contextual information in the input of an SRM is useful, its performance should surpass that of SSMO-SRM, which takes only the current step as input. This ability is referred to as the model’s capacity to assess logical coherence, meaning it can determine whether a subsequent step logically follows from the information and conclusions derived in the previous context. The results are shown in Table 1.

LLMs can be trained to evaluate logical coherence in pure mathematical language. For DeepSeek-Math-7B-Base, MO-SRM achieves an accuracy gain of +7.35% on GSM8K and +8.48% on MATH, which is higher than the gains +3.64% and 6.30% observed for SSMO-SRM. Similarly, for Qwen2-7B, MO-SRM achieves an accuracy gain of +5.31% on GSM8K and +3.94% on MATH, higher than that of SSMO-SRM +3.18% and +1.92%. This substantial difference indicates that MO-SRM, which considers the full sequence of mathematical expressions, is effective at cap-

turing logical coherence, rather than only focusing on the current step. This finding indicates that logical coherence in mathematical language can be assessed by LLMs as SRMs.

The SRMs have difficulties being trained to evaluate the logical coherence in the form of natural language. Based on our MDP definition, even after the mathematical expressions are stripped away from the current reasoning step, the natural language descriptions still include the details of the actions to be executed. In other words, the SRMs should be able to learn from these constructed preferences to identify which actions are useful for problem-solving. However, as shown in Figure 3, the dashed curves illustrate the challenges in training NT-SRMs, which were designed to evaluate the quality of the next thoughts. The training processes across various datasets and base models consistently demonstrate the difficulty in identifying preferences based solely on the descriptions of thoughts during reward training. The results presented in Table 1 further highlight the poor performance of NT-SRMs when used as scoring functions. These findings suggest that the implicit logic conveyed through natural language is difficult for LLMs to capture and evaluate effectively.

Additional Analysis

Agent & World Model	Accuracy (Gain) %	
Llama-3-70B-Instruct	GSM8K	MATH
Pass@1 (3-shots)	90.37	48.48
	(+0.00)	(+0.00)
+GS /w MO-SRM ¹	92.95	54.12
	(+2.58)	(+5.64)

Table 2: Supervise a larger model (Llama-3-70B-Instruct).

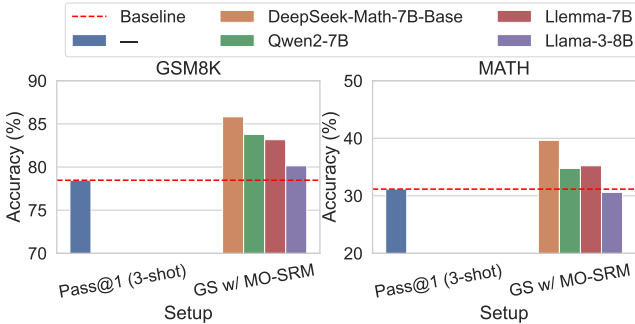


Figure 5: The performance of SRM is affected by the ability of the base model.

Supervising a larger model Despite being trained on preference data generated by a smaller model, the MO-SRM was able to effectively guide the reasoning process of a

¹The MO-SRM here is trained based on DeepSeek-Math-7B-Base with preference data generated through MCTS performed by Llama-3-8B-Instruct.

larger model and achieve substantial improvements (+2.58% on GSM8K and +5.64% on MATH) (Table 2). This further illustrates the ability of the SRMs to focus exclusively on mathematical language.

Effect of base models for MO-SRM The choice of SRM base models impacts performance (Figure 5), while this effect doesn’t appear to be entirely related to the base model’s mathematical abilities. Despite its excellent mathematical capabilities, The surprising underperformance of Llama-3-8B compared to Llemma-7B (Azerbayev et al. 2023), Qwen2-7B, and DeekSeek-Math-7B-Base, suggests that factors beyond just original mathematical ability are at play. This might be due to the challenges in self-assessment or other reasons to be explored.

Agent & World Model	Accuracy	
Llama-3-8B-Instruct	GSM8K	Accuracy
+BS w/ MO-SRM ¹		
$B = 1, c = 5$	85.82	39.64
$B = 1, c = 10$	85.90	40.06
$B = 3, c = 10$	88.17	40.24

Table 3: Effect of B and c on beam search

Effect of B and c on beam search Increasing the beam size B and the number of candidate count c will slightly improve accuracy, but this improvement will eventually plateau, as shown in Table 3.

Conclusion

Our investigation into the role of natural language and mathematical expressions in step-level reward modeling reveals that natural language descriptions are not essential for the success of these models. Through extensive experiments, we demonstrated that reward models operating solely on mathematical expressions perform comparably to those that incorporate both natural language and math. Furthermore, the difficulty in training models to evaluate the coherence of natural language thought processes underscores the challenges LLMs face in capturing implicit logical structures through language alone. We also found that the coherence of logical structure inherent in mathematical expressions can be assessed by SRMs trained based on LLMs. Given the overhead of obtaining step-level rewards, these findings offer new insights for developing more efficient and targeted reward models by isolating the most impactful components of mathematical reasoning steps.

Acknowledgments

This work was supported in part by National Key R&D Program of China, under Grant No. 2022YFC3303600 and in part by Key Laboratory of Smart Education of Guangdong Higher Education Institutes, Jinan University (2022LSYS003).

References

- Azerbayev, Z.; Schoelkopf, H.; Paster, K.; Dos Santos, M.; McAleer, S. M.; Jiang, A. Q.; Deng, J.; Biderman, S.; and Welleck, S. 2023. Llemma: An Open Language Model for Mathematics. In *The Twelfth International Conference on Learning Representations*.
- Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.
- Chen, G.; Liao, M.; Li, C.; and Fan, K. 2024a. AlphaMath Almost Zero: process Supervision without process. *arXiv preprint arXiv:2405.03553*.
- Chen, G.; Liao, M.; Li, C.; and Fan, K. 2024b. Step-level Value Preference Optimization for Mathematical Reasoning. In *Conference on Empirical Methods in Natural Language Processing*.
- Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ding, R.; Zhang, C.; Wang, L.; Xu, Y.; Ma, M.; Zhang, W.; Qin, S.; Rajmohan, S.; Lin, Q.; and Zhang, D. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*.
- Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; et al. 2024. The Llama 3 Herd of Models.
- Fedorenko, E.; Piantadosi, S. T.; and Gibson, E. A. 2024. Language is primarily a tool for communication rather than thought. *Nature*, 630(8017): 575–586.
- Feng, X.; Wan, Z.; Wen, M.; Wen, Y.; Zhang, W.; and Wang, J. 2023. Alphazero-like Tree-Search can Guide Large Language Model Decoding and Training. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.
- Hao, S.; Gu, Y.; Luo, H.; Liu, T.; Shao, X.; Wang, X.; Xie, S.; Ma, H.; Samavedhi, A.; Gao, Q.; et al. 2024. LLM Reasoners: New Evaluation, Library, and Analysis of Step-by-Step Reasoning with Large Language Models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.
- Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; and Hu, Z. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8154–8173.
- Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Li, X.; Bai, Y.; Guo, T.; Liu, Z.; Huang, Y.; Zhao, X.; Xia, F.; Luo, W.; and Weng, J. 2024. Enhancing Length Generalization for Attention Based Knowledge Tracing Models with Linear Biases. In *33rd International Joint Conference on Artificial Intelligence, IJCAI 2024*, 5918–5926. International Joint Conferences on Artificial Intelligence.
- Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let’s Verify Step by Step. In *The Twelfth International Conference on Learning Representations*.
- Ma, Q.; Zhou, H.; Liu, T.; Yuan, J.; Liu, P.; You, Y.; and Yang, H. 2023. Let’s reward step by step: Step-Level reward model as the Navigators for Reasoning. *arXiv preprint arXiv:2310.10080*.
- Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Zhang, M.; Li, Y.; Wu, Y.; and Guo, D. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Wang, P.; Li, L.; Shao, Z.; Xu, R.; Dai, D.; Li, Y.; Chen, D.; Wu, Y.; and Sui, Z. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 9426–9439.
- Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.
- Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.
- Xie, Y.; Goyal, A.; Zheng, W.; Kan, M.-Y.; Lillicrap, T. P.; Kawaguchi, K.; and Shieh, M. 2024. Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning. *arXiv preprint arXiv:2405.00451*.
- Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. 2024. Qwen2 Technical Report. *CoRR*.
- Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.
- Zhan, B.; Guo, T.; Li, X.; Hou, M.; Liang, Q.; Gao, B.; Luo, W.; and Liu, Z. 2024. Knowledge tracing as language processing: A large-scale autoregressive paradigm. In *International Conference on Artificial Intelligence in Education*, 177–191. Springer.
- Zhang, D.; Li, J.; Huang, X.; Zhou, D.; Li, Y.; and Ouyang, W. 2024. Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B. *arXiv preprint arXiv:2406.07394*.
- Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- Zheng, Y.; Li, X.; Huang, Y.; Liang, Q.; Guo, T.; Hou, M.; Gao, B.; Tian, M.; Liu, Z.; and Luo, W. 2024. Automatic

Lesson Plan Generation via Large Language Models with Self-critique Prompting. In *International Conference on Artificial Intelligence in Education*, 163–178. Springer.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; et al. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.

A Implementation Details

A.1 Prompts

System message (Agent)

You should act as a guide. You will break down the process into individual, understandable guidance step-by-step, each leading logically to the final result. I will follow your guidance by calculating the answer to each step with equations.

Your response must meet the following requirements:

1. Never say anything not related to the math problem.
2. You should not include any calculations in your instruction as that is the student’s work.
3. If the current math problem is ready to be solved by following your next guidance, start it with “Now you can answer the problem in this step.”.
4. If the final answer to the current math problem has been obtained, just say “The math problem has been solved.”

System message (World Model-GSM8K)

You are a student solving math problems under the instructions of the teacher. You should follow the step-by-step guidance posed by the teacher by calculating the answer of each step with equations until you deduce the final answer to the math problem.

Your response must meet the following requirements:

1. Never talk about anything not related to the math problem.
2. Include the equation of this step.
3. If the guidance starts with “Now you can answer the problem in this step.”, you must find the final answer to the problem in this step.
4. End with “The answer is” along with a single number to highlight the numerical (sub)answer (e.g. “The answer is 42.”).

System message (World Model-MATH)

You are a student solving math problems under the instructions of the teacher. You should follow the step-by-step guidance posed by the teacher by calculating the answer of each step with equations until you deduce the final answer to the math problem.

Your response must meet the following requirements:

1. Include the equation of this step.
2. If the subquestion is started with start it with “Now you can answer the problem in this step.”, you must find the final answer to the problem in this step.
3. You must use the LaTeX code “`boxed`” to highlight the final answer to the problem. (e.g. “ $(9 + 1)^3 = 10^3 = \boxed{1000}$ ”).

A.2 Hyperparameters

MCTS The hyperparameters of MCTS are shown in Table A.1.

Hyperparameter	Value
n (n_candidates)	6
depth_limit	8
w_{exp}	1.0
temperature (agent)	1.3
temperature (world)	0.7
n_iteration	500

Table A.1: Hyperparameters of MCTS

Step-Level Reward Modeling The hyperparameters for step-level reward modeling are shown in Table A.2.

Hyperparameter	Value
n_instances	2
gpus_per_instance	8
per_device_train_batch_size	16
gradient_accumulation_steps	2
num_train_epochs	2
warmup_ratio	0.03
learning_rate	1.41e-5
weight_decay	0.1

Table A.2: Hyperparameters of MCTS

BS w/ SRM The hyperparameters for BS w/ SRM are shown in Table A.3.

Hyperparameter	Value
n (n_candidates)	5 or 10
beam_size	1 or 3
temperature (agent)	0.7
temperature (world)	0.0

Table A.3: Hyperparameters of MCTS

A.3 Example

Greedy Search Supervised by an SRM For a better understanding of our definition of the mathematical reasoning process supervised by the SRMs, we provide an example (Figure A.1) of a greedy search, where the rewards are from the MO-SRM.

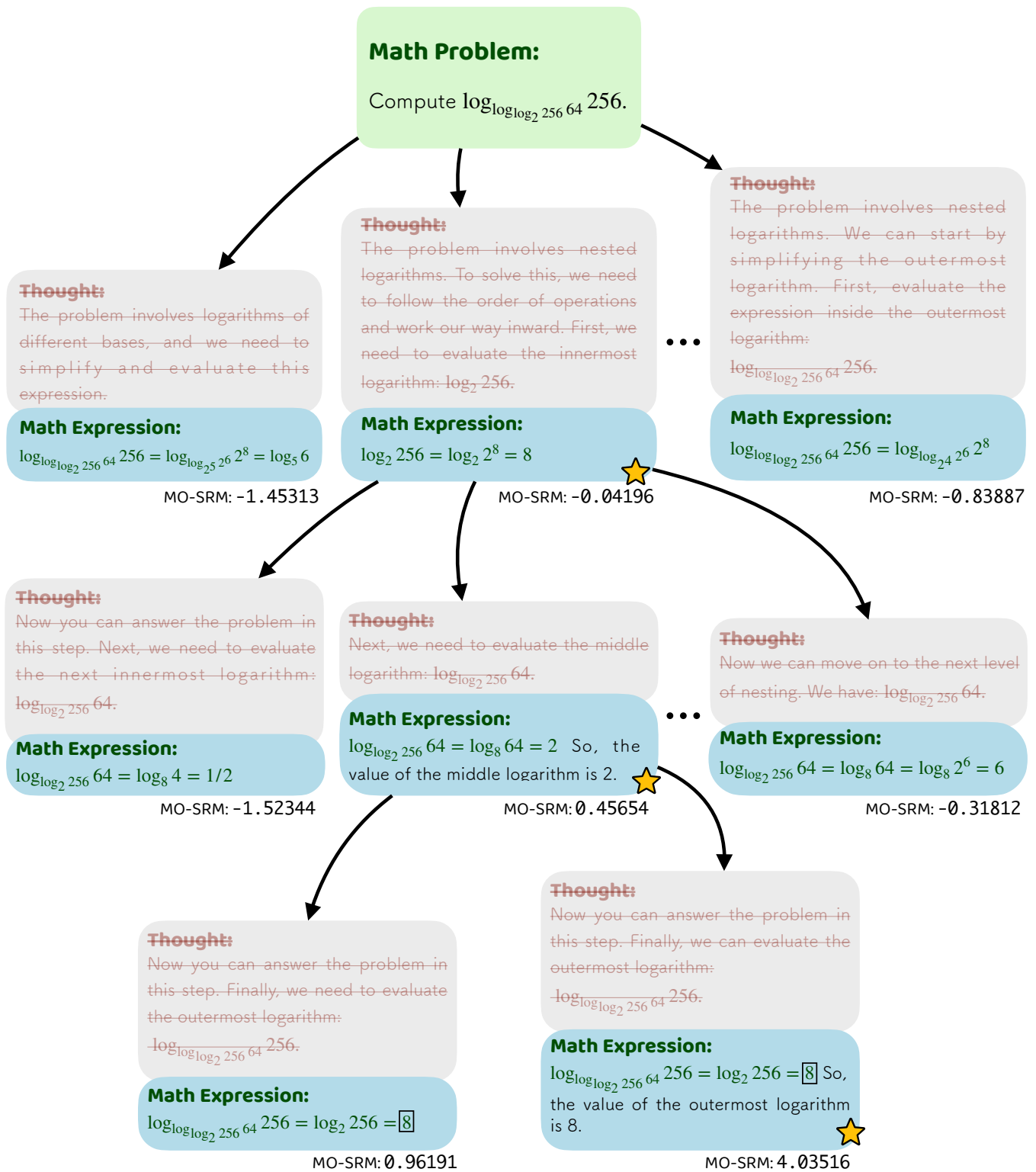


Figure A.1: An example of a case where the MO-SRM is used to supervise the GS.

A.4 Additional Results

A.5 Tendency of encouraging shorter paths

We observed that the greedy search with the SRMs tends to encourage shorter reasoning paths, although the MCTS itself does not explicitly include the path length as a preference. (Figure A.2) This observation is due to the insufficient exploitation of the MCTS process, but we need further investigation to confirm this proposition in future studies.

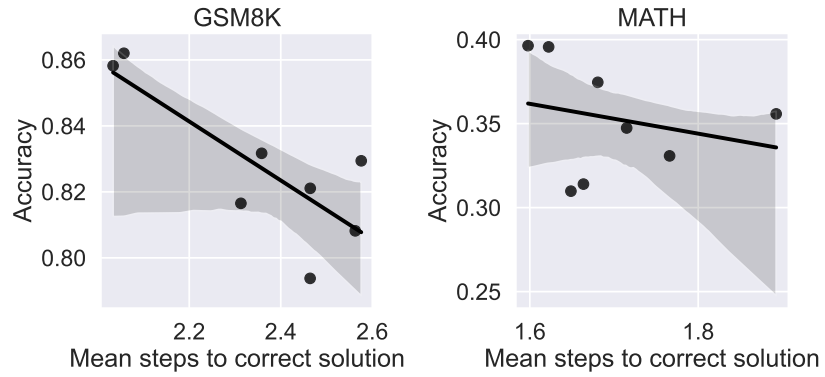


Figure A.2: Accuracy v.s. mean steps to correct solutions. Fewer steps to correct solutions tend to have higher accuracy.