Speculative Reward Model Boosts Decision Making Ability of LLMs Cost-Effectively

Jiawei Gu
Sun Yat-sen University
kuvvius@gmail.com

Shangsong Liang✉
Sun Yat-sen University
liangshangsong@gmail.com

Abstract

Effective decision-making in Large Language Models (LLMs) is essential for handling intricate tasks. However, existing approaches prioritize performance but often overlook the balance between effectiveness and computational cost. To address this, we first introduce the 3E Criteria to systematically assess the cost-effectiveness of search strategies, revealing that existing methods often trade significant efficiency for marginal performance gains. To improve LLM decision-making while maintaining efficiency, we propose the Speculative Reward Model (SRM), a plug-and-play framework that seamlessly integrates with existing search strategies. Specifically, SRM employs an external reward assigner to predict optimal actions, reducing reliance on LLMs' internal self-evaluation. And a speculative verification mechanism is used to prune suboptimal choices and guide the search toward more promising steps. We evaluate SRM on several complex decision-making tasks including mathematical reasoning, planning and numerical reasoning in specialized domains. Experimental results show that SRM reduces costs to 1/10 of the original search framework on average while maintaining effectiveness.

1 Introduction

Large Language Models (LLMs) (OpenAI et al., 2023; OpenAI, 2024; DeepSeek, 2024; Qwen, 2024) have achieved significant progress in natural language processing, excelling in text generation and comprehension (Xu et al., 2025). However, their application to complex reasoning and decision-making remains challenging (Shao et al., 2024; Zelikman et al., 2024), particularly when solving intricate problems that require structured logical inference rather than pattern-based predictions (Valmeekam et al., 2023; Shao et al., 2024).

Table 1: **Speculative Reward Models (SRM)**, a plug-and-play framework designed to balance effectiveness and efficiency. In GSM8K tasks, all paradigms followed the same setting with *GPT-3.5-turbo* and 4-shot learning. The token cost is expressed in '[Prompt Tokens]/ [Completion Tokens]'. "Ext." denotes Extensibility. For Toolchain*, which lacks direct execution capability, we estimate cost using identical prompts but exclude running time.

Paradigm	Effectiveness	Efficiency		Ext.
		Time Cost	Token Cost	
	Acc.[%]	Avg.[sec.]	Avg.[K]	
CoT(Wei et al., 2022)	70.1	3.2	0.7/0.1	✓
DFS(Yao et al., 2023)	69.9	150	70.2/5.0	✓
+ SRM	70.5	34.7	18.6/0.8	✓
BFS(Yao et al., 2023)	72.3	180	85.5/7.1	✓
+ SRM	70.1	44	22.2/1.1	✓
BS(Wan et al., 2024)	71.4	66.4	225.4/4.4	✓
+ SRM	72.3	44	30.8/1.1	✓
MCTS(Hao et al., 2023)	74.7	122.6	105.2/2.5	✓
+ SRM	**80.5**	**45.2**	**20.6/0.9**	✓
Toolchain* (Zhuang et al., 2023)	78.9	-	40.8/1.9	×

To address these limitations, early studies introduced prompting strategies to enhance reasoning, such as Chain-of-Thought (Wei et al., 2022) and AlphaZero-Like Tree-Search Method (Wan et al., 2024), which guide LLMs to generate intermediate reasoning steps to improving inference structure and accuracy. However, these methods rely solely on prompting without external validation or optimization (Song et al., 2025), limiting their reliability. Recent approaches employ tree-based search algorithms (Besta et al., 2023; Ding et al., 2023; Putta et al., 2024; Wang et al., 2024) to explore broader reasoning paths and refine intermediate steps. By systematically evaluating multiple candidates in test time scaling (Snell et al., 2024), these methods enhance both the quality and diversity of reasoning, leading to more robust decision-making.

Despite these improvements, they inevitably introduce substantial computational cost. In Table 1, we utilize our proposed ***3E Criteria—Effectiveness***, ***Efficiency***, and ***Extensibility*** to assess the cost incurred during LLM inference. *Effectiveness* repre-

✉Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 4–21

sents the success rate, *Efficiency* denotes the time and token cost, and *Extensibility* is the adaptability to new tasks.

The results reveal that existing methods offer limited performance gains at disproportionately high costs. For example, ToT (Yao et al., 2023), which employs Depth-First Search (DFS), Breadth-First Search (BFS), provides marginal performance improvements (0-3%), but incurs a 50-60× in time cost and a 100-120× escalation in inference complexity. Similarly, RAP (Hao et al., 2023) leverages Monte Carlo Tree Search (MCTS), yielding a modest performance improvements of 4-5% at the expense of a 150-300× increase in inference cost. Additionally, Toolchain* (Zhuang et al., 2023) and reasoning enhanced models like QwQ (QwenTeam, 2024), constrained by task-specific heuristics, fails to reduce cost effectively and lacks extensibility. **In this work, we seek to address:**

Research Question

How to improve the reasoning ability of LLMs while maintaining a balance between effectiveness, efficiency, and extensibility?

Inspired by studies (Huang et al., 2023) emphasizing the need for external validation in decision-making, we propose **Speculative Reward Models (SRM)**, a plug-and-play framework designed to balance effectiveness and efficiency (Jahan et al., 2016). SRM introduces external rewards to mitigate ineffective decision-making in a speculative manner (Xu et al., 2024; Chen et al., 2023; Xia et al., 2023). It consists of two key components: (1) SRM, an independent reward model that assigns scores based on decision consistency and goal alignment. (2) Speculative Verification, a mechanism that ranks candidate steps by evaluating the consistency between internal rewards from LLMs and external rewards from SRM, enabling efficient pruning of suboptimal choices and guiding the search toward more promising states, thereby reducing computational cost.

We first train SRM on datasets with weak process rewards and then fine-tune it to SRM$^+$ using strong search rewards. This allows us to provide potential success probabilities for specific steps as external reward signals to LLMs during the search phase. Extensive validation has demonstrated that our approach significantly lowers the cost to a fraction of the original search framework's, without sacrificing effectiveness. In summary, our contributions are as follows:

(1) Efficiency. The SRM framework we proposed dramatically increases efficiency with a notable reduction in cost, requiring only about 1/10 of the original search paradigms.

(2) Effectiveness. There is no sacrifice of effectiveness for SRM; in fact, by integrating reward signals for process supervision, it achieves a up to a 10% performance improvement over CoT and approximately a 2% increase compared to using searching algorithms only.

(3) Extensibility[1] SRM provides generalizable weak rewards and a universal framework for deriving strong rewards. Fine-tuning with strong rewards transforms SRM into SRM$^+$, enabling domain-specific adaptation without full retraining.

2 Problem Formulation

The decision-making process can be formulated as a Markov Decision Process (MDP) (Puterman, 1990), where the state space $\mathcal{S}$ represents all possible problem states with $s \in \mathcal{S}$, and the action space $\mathcal{A}$ consists of actions $a \in \mathcal{A}$ that transition the state toward a solution. The LLM acts as a generator $\mathcal{G}$, producing candidate actions $\mathcal{G}(a|s, prompt_1)$ and determining state transitions $\mathcal{G}(s'|s, a, prompt_2)$. A reward function $\mathcal{R}(s, a)$ evaluates the effectiveness of actions in progressing toward the goal.

Tree-based search paradigms in LLMs decompose complex problems into a sequence of manageable sub-problems, each represented as an action modifying the current state toward the final solution. The search tree $\mathcal{T} = (\mathcal{S}, \mathcal{A})$ in Figure 1 represents the decision process, where nodes are states and edges are actions. Starting from an initial state s_0, LLM iteratively generates candidate actions $A_n = \{a_n^i\}_{i=1}^K$, assigns rewards $r_{a_n^i} = \mathcal{R}(s_n, a_n^i)$, selects the optimal action a_n^*, and transitions to the next state s_{n+1}. The search process continues until the goal state s_g is reached, optimizing the cumulative expected reward along the way.

3 Method

In this section, we introduce our **SRM** framework across three key dimensions: (1) Speculative Reward (SR) for ***Efficiency***, reducing computational cost by pruning less promising search paths; (2) Reward Consistency (RC) for ***Effectiveness***, ensuring stable and reliable decision-making by aligning

[1] Refers to whether the method requires retraining to adapt to new problems across different domains.

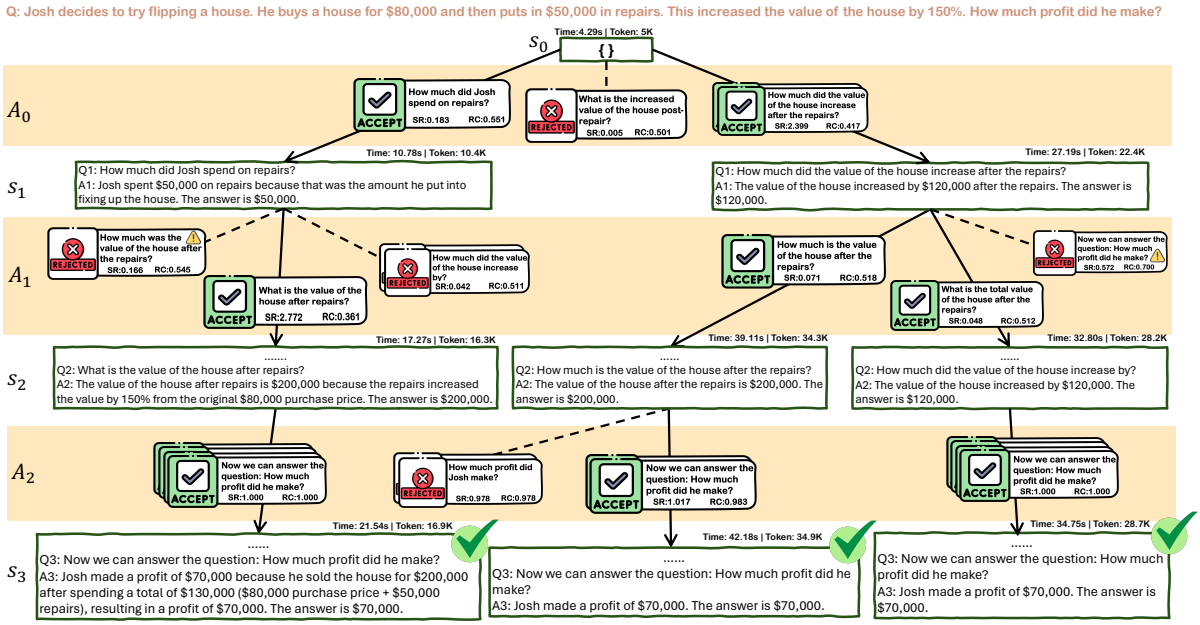

Figure 1: An example in GSM8K ($K = 4, N = 5$), where our SRM uniquely solves the case correctly across all baselines in 10 tests while achieving the lowest time and token costs. The decision-making process showcases SRM's pruning via Speculative Reward (SR), with green actions for acceptance and red for rejection. By SR, searching bypasses bad nodes and expands promising ones first. The selection strategy is determined by Reward Consistency (RC), prioritizing high-RC actions for earlier development, streamlining the path to the goal. 'Dangerous' sub-questions, characterized by excessively large spans (⚠), are pruned efficiently.

internal and external reward signals; (3) SRM$^+$ for ***Extensibility***, enabling adaptation to diverse tasks with minimal retraining.

Speculative Reward for *Efficiency* Search strategies typically rely on invoking LLMs to evaluate each state-action pair (s, a), determining the reward $\mathcal{R}(s, a)$. While effective, frequent LLM calls across large search spaces introduce significant inefficiencies. Inspired by Speculative Sampling (Xu et al., 2024; Chen et al., 2023), which accelerates inference by using a smaller model to *speculate* a larger model's predictive distribution, we propose the **SRM** to mimic the LLM as a reward assigner.

Given a pre-order state node s_n, and K candidate actions $A_n = \{a_n^1, \ldots, a_n^K\}$ generated from the LLM Generator $\mathcal{G}(\cdot)$, SRM assigns a speculative reward $\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^i)$ for each action a_n^i as:

$$\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^i) = P_\theta(a_n^i | s_n, prompt_1), \quad (1)$$

where θ is the parameters of SRM.

By bypassing LLMs for reward assignment, SRM significantly accelerates the search process. To maintain alignment with LLMs priors, following Chen et al. (2023), the reward $\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^i)$ for a_n^i is accepted with probability:

$$\min\left(1, \frac{\bigoplus(P_{\text{LLM}}(a_n^i | s_n, prompt_1))}{\bigoplus(\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^i))}\right), \quad (2)$$

where $\bigoplus(\cdot)$ denotes the normalization operator:

$$\bigoplus(f(x)) = \frac{f(x)}{\sum_x f(x)}. \quad (3)$$

Notably, $P_{\text{LLM}}(a_n^i | s_n, prompt_1)$ is directly obtained from the generation process of a_n^i, eliminating additional LLMs queries. Once the action a_n^i is accepted, we update $a_n^* \leftarrow a_n^i$ and transition to the next state s_{n+1} by $\mathcal{G}(s_{n+1} | s_n, a_n^*, prompt_2)$.This process is repeated for a_{n+1} until either the goal conditions are met or the search reaches the depth limit. If all actions $a_n^i (i = 1, 2, \cdots, K)$ are rejected, we regenerate a new candidate action set A_n' from Generator $\mathcal{G}(\cdot)$ and repeat the above process (See Algorithm 1).

Reward Consistency for *Effectiveness* Given the speculative property of the ratio in Equation 2, we define it as the Speculative Reward (SR), a key metric in our algorithm for pruning. However, assessing absolute performance alone is insufficient, the consistency of reward signals must also be considered. To this end, we propose Reward Consistency (RC) as a selection criterion, quantifying the

alignment between internal generator rewards and external SRM rewards. It is defined as:

$$RC = \frac{1}{1 + |SR - 1|} \in [0, 1]\,. \tag{4}$$

An RC value of 1 indicates complete consistency between internal and external reward signals. Their role within our SRM framework are illustrated in Figure 1. Ultimately, the cumulative reward across states (or nodes) is computed by $R_{\text{accumulated}} = SR^{\alpha} \cdot RC^{(1-\alpha)}$ where α is a hyperparameter that balance the significance of SR and RC.

SRM Training and Fine-tuning The SRM is trained on weak reward labels for each reasoning step—positive, negative, and neutral (see Appendix A.2.1 for details). Specifically, it is optimized using a cross-entropy loss function to distinguish the more advantageous action among candidates:

$$\begin{aligned} \text{loss}(\theta) = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(s_n, a_n^i, a_n^j) \sim D} \\ \left[\log\left(\sigma(\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^i) \;\; - \mathcal{R}_\theta^{\text{SRM}}(s_n, a_n^j))\right)\right], \end{aligned} \tag{5}$$

where $\mathcal{R}_\theta^{\text{SRM}}(s_n, a_n)$ represents the scalar reward assigned by SRM for preorder state s_n and available action a_n, parameterized by θ. The model favors actions that lead toward the solution, assigning them higher rewards and the dataset D contains process-supervised reward or tree-based search reward. This training approach leverages differences in weak rewards to guide SRM in quantifying the intuitive preference for actions that move toward the goal state, thereby enhancing its ability to evaluate the potential success of reasoning steps. Following (Ouyang et al., 2022), all $\binom{K}{2}$ comparisons from each prior state s_0 are processed efficiently as a single batch element to mitigate overfitting.

SRM$^+$ for *Extensibility* SRM$^+$ is fine-tuned from SRM with same loss described in Equation 5, but with a distinct *RewardTuning* dataset. This dataset includes step-level, strong rewards with specific values derived from tree-based search techniques for targeted tasks. Thus, at this stage, SRM$^+$ is more accurate to learn the relative quality of movements through strong labels. The evolution from SRM to SRM$^+$ is illustrated in Figure 2. Besides, further details on the training and fine-tuning methodologies are available in Appendix A.1, with data collection for the *RewardTuning* dataset detailed in Appendix A.2.2.

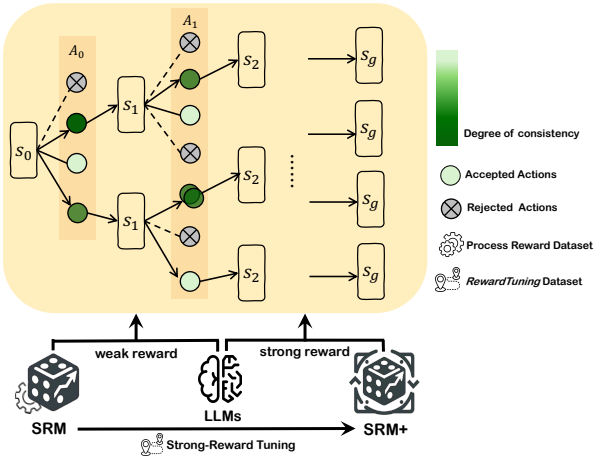

Figure 2: SRM was trained using the *PRM800K* dataset, in conjunction with LLMs, to provide weak Speculative Rewards (SR) for each action. Subsequently, SRM$^+$ underwent fine-tuning with the *RewardTuning* dataset, enabling it to generate strong SR for task-specific actions. Various actions are denoted by dots, with the intensity of their green hue indicating the magnitude of the Reward Consistency (RC) on each accepted node. A deeper green signifies a larger RC.

4 Experiment

In this section, we demonstrate the superiority of the SRM framework [2] in terms of Efficiency, Effectiveness, and Extensibility through comprehensive experiments. We evaluate SRM across a diverse range of decision-making scenarios, including mathematical reasoning on GSM8K (Cobbe et al., 2021), reasoning and planning in BlocksWorld (Valmeekam et al., 2023), and financial numeric reasoning on FinQA (Chen et al., 2021). Table 5 concisely aligns the three tasks with the decision-making problem framework.

4.1 Experiment Setup

As shown in Figure 1, we set $K = 4$ (number of candidate actions per step) and $N = 5$ (maximum search depth) for all tasks in our experiments. A detailed discussion of the GSM8K task is presented, while further information on BlocksWorld and FinQA, including their setups and case studies, can be found in Appendix C. Details regarding implementation specifics like SRM configuration, baseline alignment, and our selection of *DeBERTa-v3-large* as the base model are provided in Appendix A. Moreover, prompts used in each task are available in Appendix E.

Table 2: The result we tested 10 times on GSM8K and put on the average accuracy and cost. The values of total running time and total token cost are represented as multiples of the CoT row's value.

Method	LLaMA-2-70B			LLaMA-33B			LLaMA-2-13B		
	Effe. [Acc.]	Time [×CoT]	Token [×CoT]	Effe. [Acc.]	Time [×CoT]	Token [×CoT]	Effe. [Acc.]	Time [×CoT]	Token [×CoT]
CoT	0.54	1.0	1.0	0.29	1.0	1.0	0.20	1.0	1.0
DFS	0.52	28.4	1727.2	0.25	19.4	610.9	0.19	350.7	1306.8
+ SRM	0.54 (↑)	4.2	233.3	0.26 (↑)	2.9	32.0	0.20 (↑)	43.9	64.6
+ SRM$^+$	0.55 (↑)	4.2	241.2	0.28 (↑)	2.9	32.4	0.24 (↑)	42.0	69.5
BFS	0.58	36.3	1133.7	0.38	37.8	237.8	0.23	368.5	661.5
+ SRM	0.55	3.4	133.9	0.35	2.1	41.5	0.23	19.5	48.5
+ SRM$^+$	0.59 (↑)	3.4	123.4	0.38	2.2	42.2	0.26 (↑)	19.2	42.2
MCTS	0.61	1145	295.1	0.49	74.6	108.1	0.30	61.2	180.7
+ SRM	0.62 (↑)	8.0	66.7	0.49	2.2	19.9	0.27	15.3	33.0
+ SRM$^+$	0.64 (↑)	8.0	63.4	0.51 (↑)	2.3	20.7	0.29	15.3	31.8

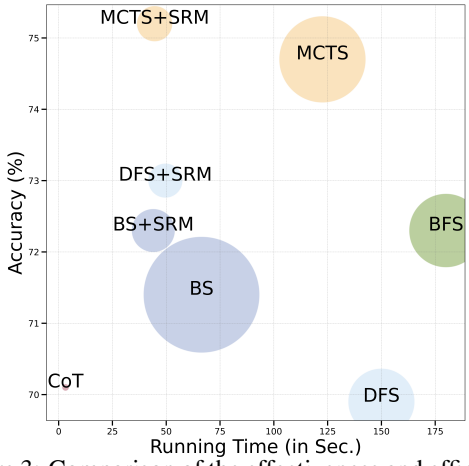

Figure 3: Comparison of the effectiveness and efficiency of search methods using the plug-and-play SRM framework. The bigger the dot is, the larger the token cost. After applying the SRM framework, it is obvious that the running time of the point representation is reduced (←), and the accuracy is flat or increased (↑).

4.2 *Effectiveness* and *Efficiency* Analysis

To evaluate the impact of SRM on effectiveness and efficiency, we present results on GSM8K from GPT-3.5-turbo and the LLaMA series (Touvron et al., 2023; Grattafiori et al., 2024) in Table 1 and Table 2. The results show that SRM significantly reduces both time and token costs by nearly 90% while maintaining or improving performance (Figure 3). Notably, these benefits come without compromising extensibility.

SRM applied to LLaMA-2-70B improves accuracy by 2% on ToT-DFS and 1% on RAP-MCTS. When used with GPT-3.5, its cost is only 10% to 30% of the original search algorithms. However, results highlight the instability of search paradigms in decision-making tasks. DFS, for example, performs 2% worse than CoT alone. Integrating DFS with SRM mitigates this decline by pruning weak nodes and expanding stronger ones. The fine-tuned SRM$^+$ further enhances search performance while stabilizing the framework at a lower cost. Additionally, SRM can be fine-tuned using other tree-based search rewards, as discussed in Appendix D. Overall, MCTS+SRM proves to be the most cost-effective approach across GPT-3.5-turbo and the LLaMA series. Among the evaluated search paradigms, **MCTS exhibits the highest accuracy yet the highest time cost.** This can be attributed to its more reliable reward system, derived from multiple simulations, rather than the self-evaluation and positional relationship utilized by BFS and DFS. Therefore, in our experiment, we use the MCTS reward in *RewardTuning* as the strong reward label to acquire SRM$^+$. Overall, MCTS+SRM emerges as the most cost-effective approach for decision-making tasks, as demonstrated using GPT-3.5-turbo and the LLaMA series.

Case Study *SRM mitigates error propagation by prioritizing reliable search paths and pruning error-prone branches.* Figures 1 and 6 compare MCTS+SRM and MCTS alone, demonstrating how SRM reduces early mistakes that would otherwise propagate through later steps. SRM prioritizes concise sub-questions with higher SR and RC, effectively pruning unreliable branches and guiding search toward more reliable paths. In contrast, MCTS alone struggles to avoid error-prone branches, leading to early mistakes that propagate through later steps. MCTS relies on fast rewards and LLM self-evaluation, which, while efficient in some cases, often fails to prevent accumulating errors. Without external supervision, minor mistakes can significantly impact tree search algorithms, as LLMs struggle to self-correct. As shown in Figures 1 and 6, reducing step size and verifying each step prevents errors from compounding, demon-

[2]Code available at: https://github.com/Kuvvius/Speculative-RM

Table 3: The baseline is MCTS. Sampling refers to the rejection sampling strategy outlined in Section 3, absent which there is no pruning. Consistent with earlier sections, token costs are denoted as [Prompt Tokens]/[Completion Tokens].

Method	Effectiveness	Efficiency	
		Time Cost	Token Cost
	Acc.[%]	Avg.[Sec.]	Avg.[K]
MCTS	74.7	122.6	105.2/2.5
+ SR + sampling	$70.2_{\downarrow 4.5\%}$	28.3	16.3/0.4
+ RC + sampling	$71.4_{\downarrow 3.3\%}$	96.5	53.2/1.5
+ $SR^{\alpha} \cdot RC^{(1-\alpha)}$ + sampling	$80.5_{\uparrow 5.8\%}$	45.2	20.6/0.9
+ SR no sampling	$78.4_{\uparrow 3.7\%}$	105.1	70.8/2.1
+ RC no sampling	$73.3_{\downarrow 1.4\%}$	143.2	98.1/2.7
+ $SR^{\alpha} \cdot RC^{(1-\alpha)}$ no sampling	$75.1_{\uparrow 0.4\%}$	58.8	34.7/0.9

strating SRM's role in stabilizing search efficiency while maintaining accuracy.

Ablation Study We conduct ablation studies with the MCTS paradigm to evaluate the impact of reject sampling via SR and selection mechanisms via RC (Table 3). The results indicate that both components in SRM's speculative approach contribute to reducing cost while maintaining performance. Using only SR for $R_{accumulative}$ significantly lowers cost but also reduces effectiveness. In contrast, relying solely on RC results in a smaller accuracy drop but at the expense of efficiency. Without sampling, cost increases due to the lack of tree pruning, sometimes exceeding the baseline search algorithms. These findings confirm SRM's effectiveness in optimizing tree-based search performance.

4.3 *Extensibility* Analysis

Table 4: Result of Blocksworld (LLaMA-2-70B) and FinQA (GPT-3.5 and GPT-4).

Mode	Method	Eff.	Time	Token
BW(Easy)	CoT	0.08	1.0x	3.8
	MCTS	0.66	560.9x	366.0
	MCTS + SRM	0.66	54.4x	40.1
	MCTS + SRM$^+$	0.68	58.3x	47.0
BW(Hard)	CoT	0.05	1.0x	3.8
	MCTS	0.51	709.5x	416.7
	MCTS + SRM	0.49	54.8x	34.2
	MCTS + SRM$^+$	0.54	69.9x	45.5
FinQA (GPT3.5)	CoT	0.49	4.5	3.4
	MCTS	0.60	160.6	200
	MCTS + SRM	0.65	51.9	54.2
	MCTS + SRM$^+$	0.68	52.1	53.7
FinQA (GPT-4)	CoT	0.70	4.9	3.5

Table 4 highlights SRM's adaptability across decision-making tasks. In Blocksworld (BW), CoT with LLaMA-2-70B struggles with planning, while MCTS improves decisions at high computational cost. SRM reduces inference by 7% while maintaining accuracy, and SRM$^+$ further enhances performance via *RewardTuning* (See Appendix A.2.2).

Beyond planning, SRM seamlessly transfers to FinQA, improving accuracy by 5% with minimal retraining, while SRM$^+$ achieves an 8% gain. Notably, SRM$^+$ enables GPT-3.5 to match GPT-4 in efficiency, demonstrating its ability to optimize LLMs across domains. By integrating speculative verification and fine-tuning with task-specific rewards, SRM ensures efficient, cost-effective adaptation to new tasks.

5 More Discussion

Diversity and randomness bring stable improvement. The methods related to Decision-making agents would have unstable issues and strongly depend on the general ability of the base model. During the reasoning process, MCTS introduces a degree of randomness in generating the final results. This randomness, combined with the diversity at intermediate nodes, allows for stable optimization of the sampling outcomes from language models. Consequently, MCTS consistently demonstrates superior performance compared to other search methods.

External signals can effectively supervise the generation process of the content. When a decision-making agent engages in complex reasoning and problem-solving, it heavily relies on the generative capabilities of the language model. However, using only self-evaluation methods often fails to provide stable and reliable judgments, making effective process supervision difficult. In such cases, introducing an external verifier for process supervision proves to be effective. The verifier can provide feedback on the quality of the model's current outputs and offer guidance, which helps improve performance.

By leveraging diversity (note that the "diversity" here differs from "diversity" in the field of information retrieval (Liang et al., 2017; Liang, 2019)) and randomness, the use of effective external signals for proper guidance can help avoid the high costs associated with repetitive exploration in the search space. Specifically, the verification signals provided by our proposed SRM in domain-specific problems, combined with search methods that **allow for sufficient exploration and randomness**, can achieve cost-effective performance improvements.

Why a relatively small model can help large base model? Our reward model underwent training that supervised the decision-making process, but it's significantly smaller compared to the generative language models it supports. The feasibility of using a smaller-scale reward model to effectively assist a much larger, more powerful model lies in our acknowledgment of the errors inherent in the weak labels provided by the Supervised Reward Model (SRM). However, within our framework, we do not intend for the more robust model to learn or replicate these errors. Instead, our aim is to guide it toward understanding the intentions behind the supervision (i.e., signals of external oversight), not the inaccuracies themselves. We maintain the assumption that the larger, base model inherently possesses all necessary reasoning and decision-making capabilities but might not currently exhibit them due to limitations in the decision-making context. Under the guidance of a weaker model, it becomes possible to activate this latent knowledge and adjust the base model towards a direction of self-reward, thereby enhancing its performance and decision-making processes in alignment with the supervisors' intentions.

6 Related Work

6.1 Decision-Making Agents

LLM-based decision-making agents, such as XoT (Ding et al., 2023), and Quiet-STaR (Zelikman et al., 2024) generate structured actions using formal languages like PDDL or API calls. These models rely on binary or scalar feedback for policy optimization, differing from human decision-making (Zhuge et al., 2025). Memory-enhanced methods (Shinn et al., 2023; Zhuang et al., 2023) treat LLMs as autonomous agents, but reward interpretation remains a challenge (Song et al., 2025). Our SRM addresses these limitations with a structured, cost-effective decision-making approach.

6.2 Tree-Based Search Algorithms

Tree-based search, including DFS, BFS, and MCTS, plays a key role in LLM-driven decision-making (Snell et al., 2024). DFS and BFS explore solutions systematically, while MCTS improves decision quality via random sampling. However, methods like ToT (Yao et al., 2023), RAP (Hao et al., 2023) and AlphaZero-Like Tree-Search Method (Wan et al., 2024) incur high inference costs due to frequent LLM calls.

6.3 Speculative Sampling

Speculative sampling (Xu et al., 2024; Chen et al., 2023; Xia et al., 2023) speeds up LLM inference by drafting candidate tokens and verifying them with a target model, reducing latency while maintaining quality. Inspired by this, SRM applies speculative verification to decision-making, using rejection sampling to prune search paths, minimize redundancy, and improve efficiency.

7 Conclusion

We propose the Speculative Reward Model (SRM), a cost-effective framework that enhances LLM decision-making by speculating on potential rewards. SRM reduces ineffective decisions through Speculative-Verification, efficiently ranking steps by given scores. Our contributions include significant cost reductions, a 10% performance improvement over CoT, a 2% increase over search-based algorithms, and broad applicability. Additionally, we introduce *RewardTuning*, a dataset for fine-tuning the reward model on three tasks. As to future work, we intend to extend our model for other tasks (Xian et al., 2025; Pasupat and Liang, 2015).

Limitations

Dependency on External Models SRM need to fine-tuned with task reward data to improve the corresponding performance on the specific task. relies on external reward models, which might introduce additional complexity and potential inaccuracies if the external models are not well-calibrated or if they fail to capture the nuances of the specific tasks.

Scalability Challenges While SRM reduces costs and improves efficiency, it is itself a relatively small model with only about 500M parameters. This limited capacity can pose challenges when scaling to more complex tasks or larger datasets, potentially hindering its ability to generalize effectively.

Acknowledgments

This work has been under development for an extended period and has benefited enormously from ongoing refinement and feedback. We are profoundly grateful to Guanzheng Chen for his invaluable guidance, insightful discussions, and unwavering support at every stage of this project. We also wish to thank Jiahao Song for generously providing

the critical resources and infrastructure, especially during the early phases, that made our implementation possible. Their contributions have been instrumental in shaping and advancing this research.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek. 2024. Deepseek-r1-lite-preview: Unleashing supercharged reasoning power. https://api-docs.deepseek.com/news/news1120. Accessed: 2024-12-29.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Ali Jahan, Kevin L Edwards, and Marjan Bahraminasab. 2016. *Multi-criteria decision analysis for supporting the selection of engineering materials in product design*. Butterworth-Heinemann.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*.

Shangsong Liang. 2019. Collaborative, dynamic and diversified user profiling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4269–4276.

Shangsong Liang, Emine Yilmaz, Hong Shen, Maarten De Rijke, and W Bruce Croft. 2017. Search result diversification in short text streams. *ACM Transactions on Information Systems (TOIS)*, 36(1):1–35.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela

Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. [Accessed 19-09-2024].

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.

Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

QwenTeam. 2024. Qwq-32b: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training. In *Forty-first International Conference on Machine Learning*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024. Chain-of-table: Evolving tables in

the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, Singapore. Association for Computational Linguistics.

Ziting Xian, Jiawei Gu, Lingbo Li, and Shangsong Liang. 2025. Molrag: unlocking the power of large language models for molecular property prediction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

Han Xu, Jingyang Ye, Yutong Li, and Haipeng Chen. 2024. Can speculative sampling accelerate react without compromising reasoning quality? In *The Second Tiny Papers Track at ICLR 2024*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. Toolchain*: Efficient action space navigation in large language models with a* search. *arXiv preprint arXiv:2310.13227*.

Mingchen Zhuge, Changsheng Zhao, Dylan R Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, et al. 2025. Agent-as-a-judge: Evaluating agents with agents.

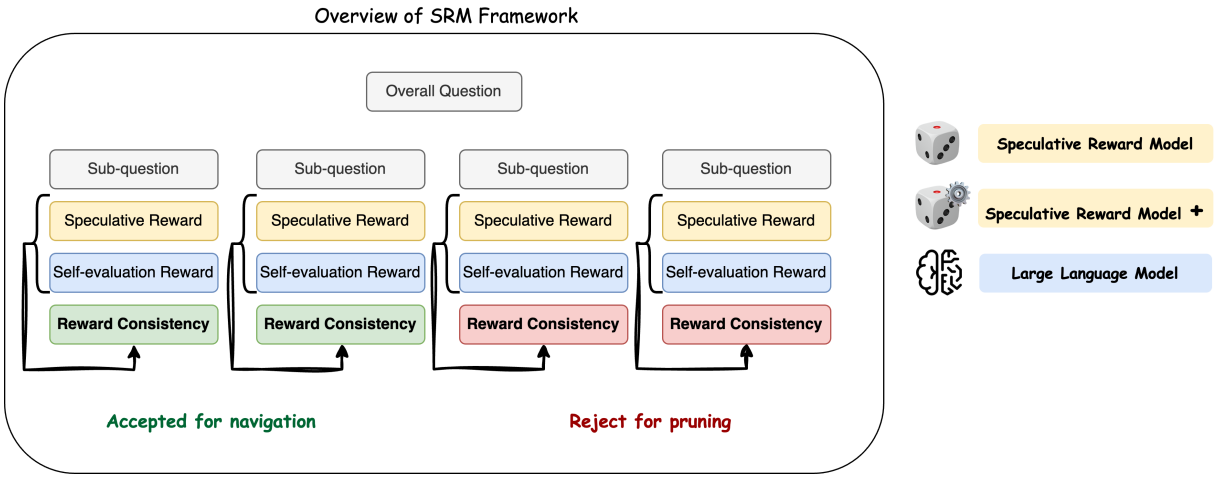

Figure 4: Example of an efficient selection process

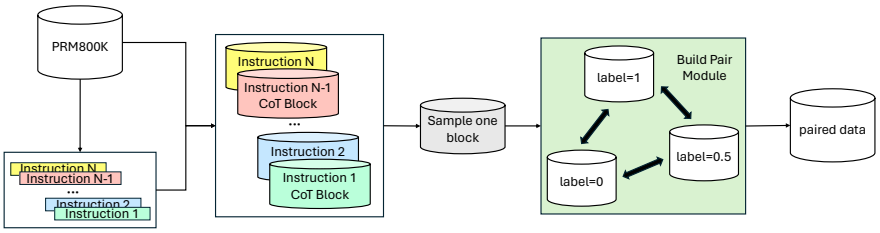

Figure 5: The process of building our weak reward dataset from *PRM800K* dataset, which SRM was trained on. The data samples of state and action pairs can be found in Appendix A.2.1.

A Implementation Details

To better illustrate the Decision-making process with SRM, we provide pseudo-code in Algorithm 1 and a selection process (including rejection for pruning and acceptance sorting for efficient navigation) as shown in the Figure 4.

A.1 LLM Configuration

In order to align the existing experimental results, we opted for the GPT-3.5-turbo (a previous version) as the engine in constructing the LLM-based agent framework. We configured the solution generation to have a maximum length of 512, with a temperature setting of 0.8, as detailed in Section 4. In the case of LLaMA-2 experiments, we similarly set the maximum solution length at 512 and the temperature at 0.8. The experiments were conducted using 8 NVIDIA Tesla V100 32GB GPUs to facilitate the inference process for both the LLaMA-2 7B and 13B models.

To maintain consistency with the established search algorithms, we adjusted weights as the same as them.

A.2 SRM Training and Fine-tuning Details

SRM was trained on DeBERTa-v3-large with sentence pairs with weak labels 7 to obtain SRM, and fine-tuned by strong labels 8 evolving into SRM+. As the loss function in Equation 5, we train SRM to learn the differences in text with different labels through comparison. Finally, with the input pairs with same state sentence, SRM can give the predicted reward labels, which show relatively good or bad. The dataset we built in our work will be fully released upon acceptance. In the A.2.1 and A.2.2, we provide further clarification and explanations through data samples.

A.2.1 Process Reward Dataset

The original training data has 1,055,517 pieces of data and 10,833 instructions (i.e. questions). After processing, there are 3,150,704 pairs. The generating process and data examples are shown in the Figure 7.

A.2.2 *RewardTuning* Dataset

We use the existing searching method to acquire the strong reward label for each step of sub-question or each state for blocks as shown in Figure 8. The form of reward is an exact value. We build all

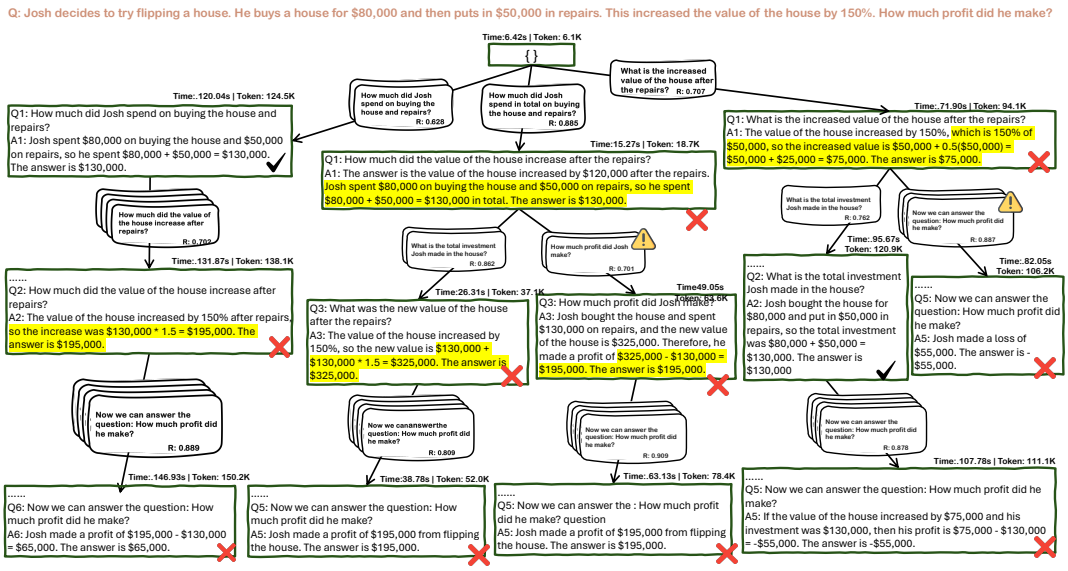

Figure 6: Bad Case of MCTS Decision-making


```
{"state": "In convex quadrilateral $ABCD$, $\\angle A = \\angle C$, $AB=CD=180$, and $AD \\ne BC$. The perimeter of $ABCD$ is 640. Find $\\cos A$.", "action": ["Let $AD=a$ and $BC=b$. So, we know that $a+b+360=640$. So, $a+b=280$.", "Let's use the Law of Cosines in the triangles $ABD$ and $BCD$.", "In $\\triangle ABD$, we have $BD^2=a^2+180^2-2(a)(180)\\cos A$.", "In $\\triangle BCD$, we have $BD^2=b^2+180^2-2(b)(180)\\cos A$.", "Now, we can cancel out $180^2$ from both sides.", "So, $a^2+180^2-2(a)(180)\\cos A=b^2+180^2-2(b)(180)\\cos A$.", "Let's subtract $a^2$ from both sides."], "action": "So, we get $-2(a)(180)\\cos A=-2(b)(180)\\cos A-a^2+b^2$.", "label": 1.0}
{"state": ["In convex quadrilateral $ABCD$, $\\angle A = \\angle C$, $AB=CD=180$, and $AD \\ne BC$. The perimeter of $ABCD$ is 640. Find $\\cos A$.","Let $AD=a$ and $BC=b$. So, we know that $a+b+360=640$. So, $a+b=280$.", "Let's use the Law of Cosines in the triangles $ABD$ and $BCD$.", "In $\\triangle ABD$, we have $BD^2=a^2+180^2-2(a)(180)\\cos A$.", "In $\\triangle BCD$, we have $BD^2=b^2+180^2-2(b)(180)\\cos A$.", "So, $a^2+180^2-2(a)(180)\\cos A=b^2+180^2-2(b)(180)\\cos A$.", "Now, we can cancel out $180^2$ from both sides.", "So, $a^2-2(a)(180)\\cos A=b^2-2(b)(180)\\cos A$.", "Let's subtract $a^2$ from both sides."], "action": "We get $-2(a)(180)\\cos A=-b^2-2(b)(180)\\cos A+a^2$.", "label": 0}
{"state": ["In convex quadrilateral $ABCD$, $\\angle A = \\angle C$, $AB=CD=180$, and $AD \\ne BC$. The perimeter of $ABCD$ is 640. Find $\\cos A$.", "Let $AD=a$ and $BC=b$. So, we know that $a+b+360=640$. So, $a+b=280$.", "Let's use the Law of Cosines in the triangles $ABD$ and $BCD$.", "In $\\triangle ABD$, we have $BD^2=a^2+180^2-2(a)(180)\\cos A$.", "In $\\triangle BCD$, we have $BD^2=b^2+180^2-2(b)(180)\\cos A$.", "So, $a^2+180^2-2(a)(180)\\cos A=b^2+180^2-2(b)(180)\\cos A$.", "Now, we can cancel out $180^2$ from both sides.", "So, $a^2-2(a)(180)\\cos A=b^2-2(b)(180)\\cos A$.", "Let's subtract $a^2$ from both sides."], "action ": "Then we get $-2(a)(180)\\cos A=-2(b)(180)\\cos A+a^2-b^2$.", "label": 0}
{"state": ["In convex quadrilateral $ABCD$, $\\angle A = \\angle C$, $AB=CD=180$, and $AD \\ne BC$. The perimeter of $ABCD$ is 640. Find $\\cos A$.", ["Let $AD=a$ and $BC=b$. So, we know that $a+b+360=640$. So, $a+b=280$.", "Let's use the Law of Cosines in the triangles $ABD$ and $BCD$.", "In $\\triangle ABD$, we have $BD^2=a^2+180^2-2(a)(180)\\cos A$.", "In $\\triangle BCD$, we have $BD^2=b^2+180^2-2(b)(180)\\cos A$.", "So, $a^2+180^2-2(a)(180)\\cos A=b^2+180^2-2(b)(180)\\cos A$.", "Now, we can cancel out $180^2$ from both sides.", "So, $a^2-2(a)(180)\\cos A=b^2-2(b)(180)\\cos A$.", "Let's subtract $a^2$ from both sides."], "action": "Now we have $-2(a)(180)\\cos A=-b^2+2(b)(180)\\cos A$.", "label": 0}
```


Figure 7: The process of generating weak reward data pairs. As the example showed, we process the data from prm800k into state and action pairs with labels

reward data for the training data set of the three task, and finally use 10% them to fine-tune our SRM$^+$. The generating process and data examples are shown in the Figure 8.

B More Analysis

Recent results on GSM8K indicate that while some methods achieve relatively high accuracy, their cost-efficiency remains a major concern. For example, QwQ (QwenTeam, 2024) achieves a very high accuracy of 93.9%. However, its time cost is not reported (denoted as "-"), and its token cost (0.7/1.6) is only slightly improved relative to baseline methods. Moreover, QwQ is marked with "×" under Extensibility, which means that despite its high performance, its applicability to new tasks is limited due to the reliance on task-specific heuristics (reasoning tasks only).

The results reveal that existing methods offer limited performance gains at disproportionately high costs. For instance, ToT (Yao et al., 2023), which employs Depth-First Search (DFS) and Breadth-First Search (BFS), provides only marginal improvements (0–3%) yet incurs a 50–60× increase in time cost and a 100–120× escalation in inference complexity. Similarly, RAP (Hao et al., 2023) uses Monte Carlo Tree Search (MCTS) to yield a modest performance improvement of 4–5%,

Algorithm 1 Decision-making process with SRM

Given candidate K actions, and depth limit of tree N.
Given Large Language Model $G(\cdot)$ as generator, and Speculative Reward Model $R(\cdot)$, action-prompt $prompt_1$ and state-prompt $prompt_2$ with few-shot examples, intial state $s_0 = \emptyset$
Initialise $n \leftarrow 0$.
while $n < N$ **do**
  for $t = 1 : K$ **do**
    Generate candidate actions auto-repressively $a_n^t \sim G(a|s_n, prompt_1)$
  end for
  Compute speculative rewards of K candidate actions respectively $a_n^t \sim R(a|s_n, prompt_2)$
  $R(a_n^1|s_n), \ldots, R(\tilde{a}_n^K|s_n)$
  for $t = 1 : K$ **do**
    Sample $\epsilon \sim U[0, 1]$ from a uniform distribution.
    if $\epsilon < \min\left(1, \frac{\bigoplus(\text{Prob}(a_n^i))}{\bigoplus(\text{Reward}(a_n^i))}\right)$ **then**
      Set $a_n \leftarrow a_n^i$ and $n \leftarrow n + 1$.
    else
      Continue
    end if
  end for
end while

but at the expense of a 150–300× increase in inference cost. Additionally, while models like Toolchain* (Zhuang et al., 2023) and reasoning-enhanced models like QwQ (QwenTeam, 2024) can achieve high accuracy, they are constrained by task-specific heuristics, fail to reduce cost effectively, and suffer from poor extensibility.

Table 1 summarizes the performance (Effectiveness), efficiency (Time and Token Cost) and extensibility of various paradigms in GSM8K tasks under the same setting with *GPT-3.5-turbo* and 4-shot learning. It is evident that despite high effectiveness, models such as QwQ, Toolchain*, and even some search-based paradigms require significant computational resources, whereas methods incorporating Speculative Reward Models (SRM) can offer a better trade-off between performance and efficiency.


```
{"state": "Georgie needs 3 avocados to make her grandmother's guacamole recipe. If she already had 5 avocados and her sister buys another 4 avocados, how many servings of guacamole can Georgie make?\n How many avocados does Georgie need to make her grandmother's guacamole recipe? Georgie needs 3 avocados to make her grandmother's guacamole recipe. The answer is 3.", "action": "How many avocados does Georgie have in the beginning?", "label": 0.6518952981160476}
{"state": "Georgie needs 3 avocados to make her grandmother's guacamole recipe. If she already had 5 avocados and her sister buys another 4 avocados, how many servings of guacamole can Georgie make?\n How many avocados does Georgie need to make her grandmother's guacamole recipe? Georgie needs 3 avocados to make her grandmother's guacamole recipe. The answer is 3.", "action": "How many avocados does Georgie already have?", "label": 0.786580977225578}
{"instruction": "Georgie needs 3 avocados to make her grandmother's guacamole recipe. If she already had 5 avocados and her sister buys another 4 avocados, how many servings of guacamole can Georgie make?\n How many avocados does Georgie need to make her grandmother's guacamole recipe? Georgie needs 3 avocados to make her grandmother's guacamole recipe. The answer is 3."], "action": "How many avocados does Georgie have?", "label": 0.7980132367124688}
```


Figure 8: The process of generating strong reward data pairs.

C Task details

Task Setup We evaluate SRM framework with the MCTS search paradigm in Blocksworld benchmark (Valmeekam et al., 2023), where the aim is to examine the framework's efficacy in guiding an agent through a sequence of actions to reorganize blocks into specified configurations. In our research, we draw from the Blocksworld dataset as outlined by (Valmeekam et al., 2023), organizing the test cases by the least number of actions they necessitate for a solution and giving four test case to prompt, as same as (Hao et al., 2023), which detailed in The plan generation task involves creating a sequence of actions to meet the goal, which showcases decision-making skills at each step of the planning process.

BW Result on Step-level Building on these results, Table 6 provides further evidence of SRM's effectiveness in both **Easy** and **Hard** modes of Blocksworld. While MCTS enhances decision-making, SRM maintains similar performance with much lower cost. In **Hard** mode, SRM$^+$ consistently improves accuracy, especially in complex tasks like the 12-step problems. These findings confirm that SRM reduces cost while preserving performance, and SRM$^+$ further extends this by improving results in more challenging scenarios.

Importantly, the set of possible actions is finite and determinable through predefined rules rather than requiring generation by an LLM. The action

Table 5: Alignment of Three Decision-making Tasks. GSM8K and FinQA, differ in complexity and domain, but both numerical reasoning tasks with action space defined by K and requiring LLM for action generation and transition. Instead, in Blocksworld, a more complex planning task, an action is composed of one of the 4 verbs (i.e., stack, unstack, put, and pick) and manipulated objects. Thus, the action set for a given state consists of m actions, with m being up to 4, generated independently of LLM assistance.

	GSM8K	**FinQA**	**Blocksworld**
Goals	Calculate the correct answer by multi-step mathematical reasoning.	Calculate the correct answer by numerical reasoning for financial problems.	Arrange the blocks into stacks on a table in the specific order.
Initial State s_0	$\emptyset$	$\emptyset$	Description of current blocks and a goal.
Goal State s_g	A correct series of problem decomposition leading to the final answer.	A correct series of problem decomposition leading to the final answer.	A feasible plan including series actions.
State s_n	All current sub-questions and answers.	All current sub-questions and answers.	Text description of the current orientation of the blocks.
Action Set A_n	K sub-questions	K sub-questions	m actions, $m \leq 4$

Table 6: Performance comparison between CoT and MCTS methods, with and without SRM, across different step sizes in Blocksworld (BW) tasks. Results are shown for both Easy and Hard modes, evaluating accuracy at 2-step, 4-step, 6-step, 8-step, 10-step, 12-step, and overall (All) steps.

Mode	**Method**	**2-step**	**4-step**	**6-step**	**8-step**	**10-step**	**12-step**	**All**
Easy	CoT	0.49	0.18	0.06	0.01	0.01	0.00	0.08
	MCTS	1.00	0.99	0.75	0.61	0.32	0.32	0.66
	MCTS + SRM	1.00	0.97	0.70	0.63	0.33	0.33	0.66
	MCTS + SRM$^+$	1.00	0.99	0.76	0.65	0.33	0.35	0.68
Hard	CoT	0.22	0.14	0.02	0.02	0.00	0.00	0.05
	MCTS	0.67	0.76	0.74	0.48	0.17	0.09	0.51
	MCTS + SRM	0.65	0.74	0.73	0.48	0.23	0.11	0.49
	MCTS + SRM$^+$	0.68	0.79	0.78	0.55	0.31	0.15	0.54

space is dynamically generated, considering both domain-specific constraints and the current orientation of the blocks. For state transitions, the framework consults a Large Language Model (LLM) to forecast the impacts of actions on the blocks' states, updating the current state to reflect new conditions and eliminate outdated ones. The LLM, in conjunction with the SRM, generates Successor Representations (SR) and Reward Contexts (RC) for potential actions, which then inform the state transition function. The process concludes once the goal state is realized or when the search hits the predetermined depth limit.

Algorithm 2 Tree-based Search in LLMs.

Input: s_0: input; G: large language model; M: the maximum exploring steps; T: the dynamic decision tree for search; $\mathcal{R}(s_n, a_n^k)$: function to return specific reward
Initialize $T = \{S, A\}$; $S \leftarrow s_0$; $A \leftarrow \emptyset$.
for $t = 1$ to N **do**
  $A_n = \{a^{(i)}\}_{i=1}^{k} \leftarrow G(s_n)$ ▷ Invoking
  $a_n^* \leftarrow \arg\max_{a_n \in A_n} \mathcal{R}(s_n, a_n)$
  Add a_n as the edge of s_n.
  $s_{n+1} \leftarrow G(s_n, a_n^*)$
  Update s_{n+1} as a node of T. ▷ Invoking
end for
Output: The goal state s_g including reasoning steps and answer.

D Tree-based search Reward

Rewards are acquired by tree-based search algorithms, different from common reward for language model (Kwon et al., 2023; Shinn et al., 2023). And all the search methods employed are unsupervised, yet they vary in the balance they strike between exploration and efficient selection.

We would like to detail three kinds of reward designs with the order of decreasing exploration. Besides, we leave the more reward settings corresponding to the algorithms in the future work. Generally, tree-based search algorithms could own their corresponding reward configure, showing the

flexibility of our framework.

D.1 Priority Reward

This type of reward are designed for the search with certain priority. Taking DFS for an example, it begins with "root" state s_0 and then iteratively choose the first candidate action a_n^1 while there are K candidate action nodes. Until it reached the depth limit or the goal state s_g containing the final correct answer. It will then proceed down the new path as it had before, backtracking as it encounters dead-ends. Besides, Self-consistancy Chain-of-Thought (Wang et al., 2022) can be expressed in reward form with majority voting as a priority.

$$\mathcal{R}_{\text{DFS}}(s_n, a_n^i) = \begin{cases} 1 & \text{if } i = \inf\{j | a_n^j \text{ not visited}\}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\inf\{j|a_n^j \text{ not visited}\}$ represents the smallest index j among all actions a_n^j that have not been visited.

D.2 Heuristic Reward

If only take confirmed priority for one-hot reward, the search process becomes aimless leading to low efficiency. Heuristic search algorithms are designed to solve the problem of search efficiency, such as Greedy Best First Search (GBFS), Dijkstra and A*. Aligned with the characteristic of algorithms, Heuristic reward defined by the heuristic function $h(s)$. Here,we would like to take GBFS for an example and list other heuristic reward in the appendix. the distance from the current state s_n to the target state s_g is used as the heuristic reward, leading the search direction correctly. Given a heuristic function $h(s)$ estimating the cost from any state s to the goal state s_g, the heuristic reward for an action a_n^i at state s_n is defined as follows:

$$\mathcal{R}_{\text{GBFS}}(s_n, a_n^i) = \begin{cases} h(s_{n+1}) & \text{if } s_{n+1} \text{ is reached by } a_n^i, \\ -\infty & \text{otherwise,} \end{cases}$$

where $h(s_{n+1})$ represents the heuristic cost from the resulting state s_{n+1}, after taking action a_n^i, to the goal state s_g. The action leading to the state with the lowest heuristic cost is preferred, guiding the search towards s_g.

D.3 Simulated rewards

With the fixed heuristic function for reward, it is evident that most of the decision space lacks coverage, resulting in insufficient exploration for searching. In contrast, simulated search algorithms like MCTS, would explore exhaustively within entire decision space. In this kind of algorithms, an iterative simulation cycle would continue until a terminal state arrived, which usually encompasses three phases: selection, expansion and backpropagation. Alongside the simulation process, a state-action value function $Q(s_n, a_n)$ is maintained, indicating the expected future reward lf taking action a_n in state s_n. To control the balance between exploration and exploitation, Upper Confidence bounds applied to Trees is often used. For each iteration of simulation, the selected action a^* should be :

$$a_n^* = \underset{a_n \in A_n}{\text{argmax}} \left[Q(s_n, a_n) + w\sqrt{\frac{N(s_n)}{1 + N(s_n, a_n)}} \right],$$

where $N(s)$ is the number of times state s has been visited in previous iterations, $N(s_n, a_n)$ is the number of times that a_n is selected at the state s_n, and weight w controls the proportion of exploration and development.

If taking MCTS as an example and supposed that to abtain the reward of an action needs simulate d times, simulated rewards can be expressed as follow:

$$\mathcal{R}_{\text{MCTS}}(s_n, a_n^i) = \frac{1}{N(s_n, a_n^i)} \sum_{k=1}^{N(s_n, a_n^i)} Q(s_n, a_n^k)\,.$$

E Prompt

For transition in SRM, we prompt:

Prompt

For each sub-question, please answer it in a complete sentence that includes your reasoning. And the last sentence ends with `"{answer_instruction}"` followed by a concise answer.

To apply CoT, we prompt:

Prompt

Q: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
A: Natalia sold 48 clips in April and half as many clips in May, so she sold $48 \div 2 = 24$ clips in May. Altogether, she sold $48 + 24 = 72$ clips. The answer is **72**.

Q: James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?
A: James writes a 3-page letter to 2 different friends twice a week, so he writes $3 \times 2 \times 2 = 12$ pages every week. There are 52 weeks in a year, so he writes $12 \times 52 = 624$ pages a year. The answer is **624**.

Q: Alexis is applying for a new job and bought a new set of business clothes to wear to the interview. She went to a department store with a budget of $200 and spent $30 on a button-up shirt, $46 on suit pants, $38 on a suit coat, $11 on socks, and $18 on a belt. She also purchased a pair of shoes, but lost the receipt for them. She has $16 left from her budget. How much did Alexis pay for the shoes?
A: Alexis spent $30 on a button-up shirt, $46 on suit pants, $38 on a suit coat, $11 on socks, and $18 on a belt, so she spent

$$30 + 46 + 38 + 11 + 18 = \$143$$

on everything else. Alexis had a budget of $200 and finally there was $16 left, so she spent

$$200 - 16 = \$184$$

in total. Since Alexis has spent $143 on everything else, she spent

$$184 - 143 = \$41$$

on the shoes. The answer is **41**.

Q: Ken created a care package to send to his brother, who was away at boarding school. Ken placed a box on a scale, and then he poured into the box enough jelly beans to bring the weight to 2 pounds. Then, he added enough brownies to cause the weight to triple. Next, he added another 2 pounds of jelly beans. And finally, he added enough gummy worms to double the weight once again. What was the final weight of the box of goodies, in pounds?
A: Ken poured jelly beans into the box until the weight was 2 pounds, so the weight of the box was 2 pounds at first. Then Ken added enough brownies to cause the weight to triple, so the weight of the box was

$$2 \times 3 = 6$$

pounds. After Ken added another 2 pounds of jelly beans, the weight of the box was

$$6 + 2 = 8$$

pounds. Finally, he added enough gummy worms to double the weight once again, so the weight of the box was

$$8 \times 2 = 16$$

pounds. The answer is **16**.

Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

A: Janet's ducks lay 16 eggs per day. She consumes

$$3 + 4 = 7$$

eggs daily, leaving her with

$$16 - 7 = 9$$

eggs to sell. Since each egg sells for $2, her total daily earnings are

$$9 \times 2 = \$18.$$

The answer is **18**.

To get the transited state for the given action and state in BW, we prompt:

Prompt

I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do:
- Pick up a block
- Unstack a block from on top of another block
- Put down a block
- Stack a block on top of another block

I have the following restrictions on my actions:
- I can only pick up or unstack one block at a time.
- I can only pick up or unstack a block if my hand is empty.
- I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.
- I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.
- I can only unstack a block from on top of another block if the block I am unstacking is clear. Once I pick up or unstack a block, I am holding the block.
- I can only put down a block that I am holding.
- I can only stack a block on top of another block if I am holding the block being stacked.
- I can only stack a block on top of another block if the block onto which I am stacking the block is clear. Once I put down or stack a block, my hand becomes empty.

After being given an initial state and an action, give the new state after performing the action.
[SCENARIO 1]
[STATE 0]
I have that, the white block is clear, the cyan block is clear, the brown block is clear, the hand is empty, the white block is on top of the purple block, the purple block is on the table, the cyan block is on the table and the brown block is on the table.
[ACTION] Unstack the white block from on top of the purple block.
[CHANGE] The hand was empty and is now holding the white block, the white block was on top of the purple block and is now in the hand, the white block is no longer clear, and the purple block is now clear.
[STATE 1]
I have that, the purple block is clear, the cyan block is clear, the brown block is clear, the hand is holding the white block, the white block is in the hand, the purple block is on the table, the cyan block is on the table and the brown block is on the table.

[SCENARIO 2]
[STATE 0]
I have that, the purple block is clear, the cyan block is clear, the white block is clear, the hand is empty, the cyan block is on top of the brown block, the purple block is on the table, the white block is on the table and the brown block is on the table.
[ACTION] Unstack the cyan block from on top of the brown block.
[CHANGE] The hand was empty and is now holding the cyan block, the cyan block was on top of the brown block and is now in the hand, the cyan block is no longer clear, and the brown block is now clear.
[STATE 1]
I have that, the purple block is clear, the brown block is clear, the cyan block is in the hand, the white block is clear, the hand is holding the cyan block, the purple block is on the table, the white block is on the table and the brown block is on the table.

[SCENARIO 3]
[STATE 0]
I have that, the red block is clear, the blue block is clear, the hand is empty, the red block is on top of the yellow block, the blue block is on top of the orange block, the orange block is on the table and the yellow block is on the table.
[ACTION] Unstack the red block from the yellow block.
[CHANGE] The hand was empty and is now holding the red block, the red block was on top of the yellow block and is now

in the hand, the red block is no longer clear, and the yellow block is now clear.
[STATE 1]
I have that, the yellow block is clear, the blue block is clear, the hand is holding the red block, the red block is in the hand, the blue block is on top of the orange block, the orange block is on the table and the yellow block is on the table.