OVM, Outcome-supervised Value Models for Planning in Mathematical Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) often struggle with maintaining accuracy throughout multiple multiple reasoning steps, especially in mathematical reasoning where an error in earlier steps can propagate to subsequent ones and it ultimately leading to an incorrect answer. To reduce error propagation, guided decoding is employed to direct the LM decoding on a step-by-step basis. We argue that in guided decoding, assessing the potential of an incomplete reasoning path can be more advantageous than simply ensuring per-step correctness, as the former approach leads towards a correct final answer. This transforms the task into a *value estimation* problem in planning.

Inspired by the findings that *outcome supervision for guided decoding essentially acts as a value model*, we propose Outcome-supervised Value Model (OVM) that employs outcome supervision for training a value model, which prioritizes steps that lead to accurate conclusions. Furthermore, the OVM eliminates the need for labor-intensive annotations of step-level correctness, thereby significantly enhancing its scalability. Our experiments on two multi-step mathematical reasoning datasets, GSM8K and Game of 24, demonstrate the superior performance of the OVM model. Notably, in GSM8K, our **OVM-7B model achieves state-of-the-art results among LLMs up to 13B parameters**; especially it does not utilize GPT-4 or code execution. These findings offer a novel perspective on the role of outcome supervision in training value models for multi-step reasoning tasks and provide theoretical justification for its advantage in value estimation for guided decoding.

1 Introduction

Multi-step reasoning problems are challenging for even large language models (LLMs) (Creswell et al., 2023; Press et al., 2022; Wei et al., 2022). Chain of Thought (CoT) outputs a series of intermediate reasoning steps before the final answer, which significantly improves the performance (Wei et al., 2022; Suzgun et al., 2023).

Verifying complete solutions Recent studies (Cobbe et al., 2021, Uesato et al., 2022, Lightman et al., 2023) have focused on training a verifier, also referred to as a '*reward* model', to verify the correctness of a complete solution among various candidates (Cobbe et al., 2021). This training generally involves two types of supervision for training: *outcome supervision* and *process supervision*. Recent research has demonstrated a clear advantage of *process supervision* over *outcome supervision* for training reward models in terms of verifying complete reasoning paths (Lightman et al., 2023).

Guided decoding during intermediate steps However, errors often happen during the decoding of intermediate steps, leading to subsequent inaccuracies due to *error propagation*. For instance, GPT-4 often struggles with the initial step in the Game of 24, yet it can solve the task with multiple attempts (Yao et al., 2023). To this end, guiding language decoding with step-level evaluation has been proposed (Xie et al., 2023; Khalifa et al., 2023; Yao et al., 2023). Paralleling the concepts of *rewards* and *values* in reinforcement learning, the criteria for step-level evaluation could be either future-agnostic (Xie et al., 2023) or future-oriented (Yao et al., 2023); the latter (i.e., value models) seems better as it has a longer-horizon perspective.

Value-based guided decoding. In line with value-based guided decoding that considers the potential of the possible future-generated solutions, the challenge lies in value estimation. Previous research primarily achieved this through extensive lookahead sampling or simulation to estimate the long-term returns (Hao et al., 2023; Zhu et al., 2023; Yao et al., 2023); this introduces an additional decoding cost during the inference of an LLM. An alternative method is to train a value model that enables value estimation during inference without the need for simulation. Inspired by

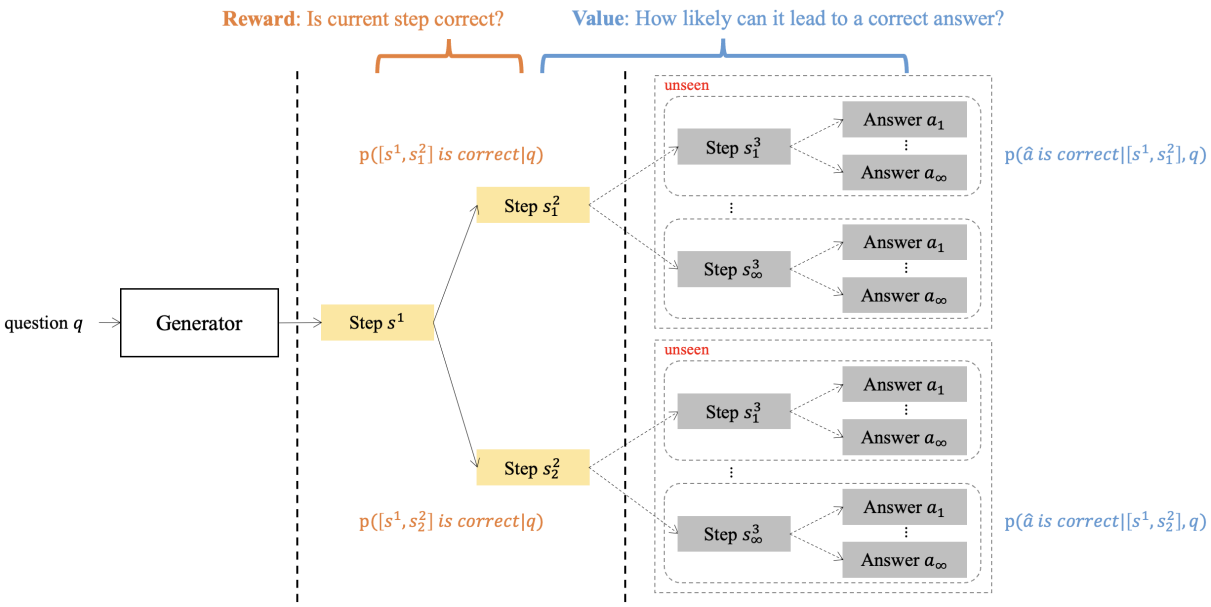


Figure 1: Distinction between reward and value. When evaluating partial paths (here for the first two steps), reward focuses on the current states, while value focuses on the (still unseen) future outcomes.

the findings that *outcome supervision for guided decoding essentially acts as a value model*, as found in this paper, we propose the use of outcome supervision to train a value model for value estimation without simulation during inference, called **OVM**.

Experiments are conducted on two popular multi-step mathematical reasoning datasets – GSM8K (Cobbe et al., 2021) and Game of 24 (Yao et al., 2023). In GSM8K, our OVM-7B model obtains state-of-the-art performance among models with up to 13B parameters, attaining a 84.7% accuracy without resorting to supplementary datasets, GPT-4, or executing programs. In Game of 24, OVM-7B reaches 78.7% success rate with merely 20 nodes visited per step, in stark contrast to its 11% greedy success rate and 11.7% with majority voting over 100 reasoning paths. Furthermore, we demonstrate that our method attains competitive, and often superior, performance using fewer sampled reasoning paths compared to complete path verification, on both GSM8K and Game of 24. This indicates the effectiveness of OVM in value estimation as a future-oriented evaluation.

In summary, our contributions are three-fold. (1) **An in-depth analysis on guided decoding**: we extend the previous discussion on outcome supervision and process supervision to the realm of guided decoding. We theoretically prove that *outcome supervision for guided decoding is secretly a value model*. (2) **A novel approach of OVM**: we propose Outcome Value Models for guided decoding that emphasize the potential correctness of the final answer rather than focusing solely on the current (partial) path's correctness. Importantly, OVM with *outcome supervision* does not need costly step-level annotations typically required by *process supervision*, making it more scalable. (3) **Significance of OVM**: OVM (7B) achieves state-of-the-art results in GSM8K among LLMs up to 13B parameters; especially it does not leverage additional data, GPT-4, or code execution.

2 Background

We give the problem definition of mathematical reasoning and guided decoding, as well as its dual paradigms (i.e., reward-based and value-based).

2.1 Problem Defintion

We first introduce the mathematical reasoning problem definition and then introduce our adopted paradigm.

Definition. A mathematical reasoning question *$q \in \mathcal{Q}$ requires a sequence of steps to be addressed, whose solution path is $S = [s^1, \ldots, s^m, a]$, where s^i represents the i-th step, m is the number of steps, and a is the final answer.*

To alleviate the issue of potential error propagation from previous steps in a single solution chain, one approach is sampling multiple steps from the generator and filtering. This is called guided decoding, which incorporates a new evaluation criterion

Notation	Description
q	Mathematical reasoning question requiring a sequence of steps
$\mathcal{Q}$	A question set
S	Solution path for a question, $S = [s^1, \ldots, s^m, a]$
s^i	The i-th step in a solution path
a	Final answer in a solution path
m	Number of steps in a solution path
y	A binary label, either 1 or 0, indicating the correctness of a
$S^{(1:t)}$	Partial solution path up to step t, $S = [s^1, \ldots, s^t]$
$\mathbb{S}^{(1:t)}$	Set of candidate partial paths $\mathbb{S}^{(1:t)} = \{S_k^{(1:t)}\}_{k=1}^K$
K	Sampling size for candidates when inference
b	Beam size for selecting top-scored candidates
n	Number of sampled paths per question for training value models
Φ	The language model as the generator
f	A scoring model that maps a partial path to a number
$v^{(1:t)}$	Value (a number) of a partial path up to step t
$\boldsymbol{\theta}$	Model parameter
PRM	Process Reward Model, trained with process supervision
ORM	Outcome Reward Model, trained with outcome supervision
OVM	Outcome Value Model, trained with outcome supervision

Table 1: Summary of Notations Used in the paper

to select steps during model generation (Yao et al., 2023; Xie et al., 2023; Khalifa et al., 2023; Feng et al., 2023).

Guided decoding Guided decoding intervenes in the generation process with a new evaluation criterion, in contrast to vanilla sampling which is solely based on the Language Model (LM) probabilities. Specifically, for each step t, the generator Φ produces K candidate paths $\mathbb{S}^{(1:t)} = \{S_k^{(1:t)}\}_{k=1}^K$ based on LM probabilities, where $S_k^{(1:t)} = [s_k^1, \ldots, s_k^t]$. Then, given an evaluation criterion $f(\cdot)$ that can score an incomplete path $S^{(1:t)}$, we can select the top-b paths, from which the generation continues

$$\left\{ S_k^{(1:t)} \middle| k \in \underset{k=1,\cdots,K}{\operatorname{argtop}_b} f(S_k^{(1:t)}; q) \right\}$$

This approach is primarily characterized by two categories of guiding criteria: reward and value, which are two concepts in reinforcement learning (Sutton and Barto, 2018), see details in Section 2.2.

2.2 Reward-based and Value-based Guided Decoding

In this subsection, we introduce 'reward'-based approaches and 'value'-based approaches for mathematical reasoning in Sec. 2.2.1 and 2.2.2

2.2.1 Reward-based Guided Decoding

Reward-based approaches (Xie et al., 2023; Khalifa et al., 2023; Hao et al., 2023), focusing on the generated steps, assess the correctness of the current steps in mathematical reasoning, i.e. $p(S^{(1:t)}$ is correct$|q)$.

Outcome supervision vs. process supervision In mathematical reasoning, reward models are well known in evaluating complete solution paths $p(S$ is correct$|q)$, also called "verifiers" (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023; Li et al., 2022; Khalifa et al., 2023). There are two supervision strategies to train a verifier, distinguished by the granularity of the supervision signals, we refer to Appendix A for training details.

Outcome Supervision simply focuses on the correctness of the final answer, at a coarser granularity. The trained model is called Outcome Reward Model (ORM) (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023).

Process Supervision offers more fine-grained, step-wise labels of the solution path, providing per-step correctness. The trained model is called Process Reward Model (PRM) (Uesato et al., 2022; Lightman et al., 2023; Li et al., 2023b).

Current research indicates that process supervision generally outperforms outcome supervision since the former adapts finer-grained supervision in *verifying complete paths* (Lightman et al., 2023). However, in guided decoding that *verifies incomplete paths*, typical reward models might overlook the current (incomplete) path's future implications, which will be further discussed in Section 2.2.2.

2.2.2 Value-based Guided Decoding

Value-based approaches (Yao et al., 2023; Hao et al., 2023; Feng et al., 2023) estimate the expected future *rewards* when starting from a given state (i.e. the current incomplete reasoning path), which is future-oriented. This is contrast to the definition of *rewards* that is determined only by the seen incomplete path and agnostic to the future path. As shown in Figure 1, reward models assess paths in a backward direction (e.g., the correctness of seen steps) while value models assess paths in a forward direction (e.g., the potential correctness the final path with additional future unseen steps and the answer $\hat{a}$ [1]). Interestingly, we could term the *value-based guided decoding* as "*planning*" based on its nature of future orientation.

[1] In reinforcement learning, value is defined as the expected cumulative reward it receives in the long run with a discount factor: $\sum_{j=1}^{m-t} \gamma^{j-1} R_{t+j}$. In our scenario, the discount factor is 1, all intermediate rewards $R_{t+1} R_{t+2} \cdots R_{m-1}$ are 0, and the final reward R_m is 1 if the answer is correct otherwise 0. So the cumulative reward is either 1 or 0 dependent on the answer correctness. Therefore, the expected cumulative reward is exactly the probability of correct answers.

3 Outcome Supervised Value Models for Guided Decoding

3.1 Motivation

Challenge of training value models Unlike labels of reward models can be annotated manually on a given (incomplete) reasoning path, it is challenging to obtain ground truth of value models for each incomplete path during guided decoding. The reason is that it is computationally-heavy to calculate the expected rewards among all possible future paths starting from the seen (incomplete) path, especially the number of resulted sequences grows exponentially w.r.t the length of reasoning paths.

Rationale behind outcome supervised guided decoding as a value model Therefore, the challenge in training value models lies in *estimating or labeling the value of observed reasoning paths*. Recalling the types of supervision – either outcome or process – it's evident that process supervision is confined to paths already seen. However, outcome supervision appears to have the potential to assess the probable correctness of resulting final paths, starting from the current incomplete one. Intriguingly, upon theoretical examination, we discover that **outcome supervision for guided decoding essentially acts as a value model**, as detailed in Sec. 3.2. This revelation has inspired the adoption of outcome-supervised value models specifically tailored for guided decoding [2] .

3.2 Outcome Supervision Leads to a Value Model for Guided Decoding

We show theoretically that, given binary labels of individual samples, outcome supervision implicitly estimates the global labels, or value, of the intermediate steps during the optimization process.

Claim. *For an outcome-supervised model $f_{\boldsymbol{\theta}}(\cdot)$ parameterized by the optimal parameter $\boldsymbol{\theta}$, its score of $S^{(1:t)}$ is the approximated probability of it reaching a correct answer, i.e.,*

$$f_{\boldsymbol{\theta}}(S^{(1:t)}; q) = p(\hat{a} \textit{ is correct} | S^{(1:t)}, q) \quad (1)$$

Proof. Suppose for each question q, we have the generator producing n solution paths $\{S_i\}_{i=1}^n$ with the corresponding answers $\{a_i\}_{i=1}^n$. The label y_i is 1 if a_i is correct otherwise 0. The mean squared error loss of outcome supervision is

$$l(S_i^{(1:t)}, y_i; q) = \left(f_{\boldsymbol{\theta}}(S_i^{(1:t)}; q) - y_i\right)^2 \quad (2)$$

[2] Feng et al. (2023) is a concurrent work with us for value model.

Given the training question set $\mathcal{Q}$, the overall objective is

$$L = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{m_i} \left(f_{\boldsymbol{\theta}}(S_i^{(1:t)}; q) - y_i\right)^2 \quad (3)$$

Denote $\mathrm{v}_x^q = f_{\boldsymbol{\theta}}(x; q)$, the partial derivation of v_x^q is

$$\nabla_{\mathrm{v}_x^q} L = \frac{1}{|\mathcal{Q}|} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{m_i} 2\Gamma(S_i^{(1:t)} = x)(\mathrm{v}_x^q - y_i) \quad (4)$$

Set $\nabla_{\mathrm{v}_x^q} L = 0$, we can see

$$\mathrm{v}_x^q = \frac{\sum_{i=1}^{n} \sum_{t=1}^{m_i} \Gamma(S_i^{(1:t)} = x) y_i}{\sum_{i=1}^{n} \sum_{t=1}^{m_i} \Gamma(S_i^{(1:t)} = x)} \quad (5)$$

which is $p(\hat{a} \text{ is correct} | x, q)$, whose estimation's precision depends on the sampling. Choose the model satisfying

$$f_{\boldsymbol{\theta}}(x; q) = \frac{\sum_{i=1}^{n} \sum_{t=1}^{m_i} \Gamma(S_i^{(1:t)} = x) y_i}{\sum_{i=1}^{n} \sum_{t=1}^{m_i} \Gamma(S_i^{(1:t)} = x)} \quad (6)$$

for all $x \in \{S_i^{(1:t)}\}$ is the optimal solution minimizing the loss function. Hence the optimal solution satisfies

$$f_{\boldsymbol{\theta}}(S^{(1:t)}; q) = p(\hat{a} \text{ is correct} | S^{(1:t)}, q) \quad (7)$$

□

Intuitive Explanation This indirect method of probability estimation in outcome supervision simplifies the value model training process, which avoids the need for explicit step-level continual sampling and estimation for training labels. Instead, it leverages the binary correctness of individual samples as training labels, forcing the optimal solution to be the probability of being correct under mean square error, which is similar to the Monte Carlo method to estimate the expectation.

3.3 Rethinking Supervision for Guided Decoding: Outcome v.s. Process

With the above discussion, outcome supervision and process supervision can be different in the context of guided decoding. We claim that outcome supervision supersedes process supervision in this scenario for two reasons.

Outcome supervision is preferred due to its inherent future-guided orientation For guided decoding, intuitively we should adopt a forward-looking approach that prioritizes the final answer's correctness over mere the current path's. This

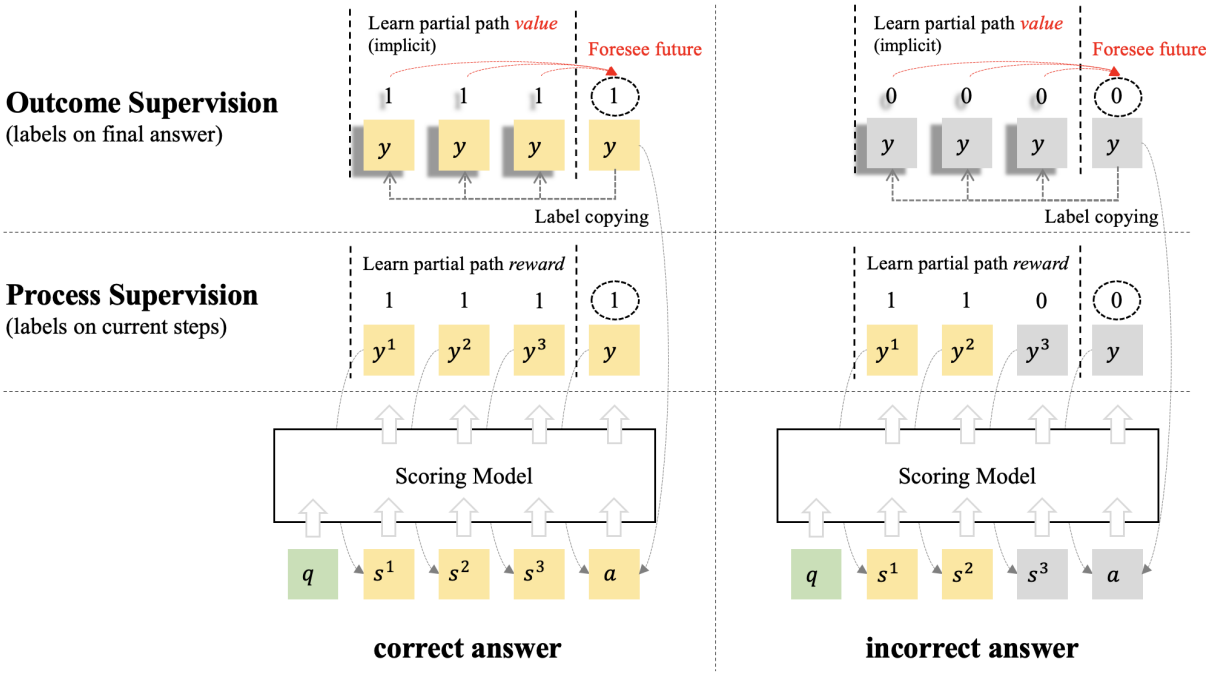


Figure 2: Outcome supervision and process supervision on training models to evaluate *complete paths*. Given a question q and a solution path $[s^1, \cdots, s^m, a]$, models are trained to predict path correctness (circled output scalar on the last token). Training is conducted at the token level for robustness. Outcome supervision replicates the final answer's correctness label across all tokens (indicated by shaded labels), while process supervision details per-step correctness labels. For *partial path evaluation*, process-supervised models learn to predict step-level correctness, i.e. reward. Notably, outcome-supervised models implicitly learn to foresee the future, predicting values, due to labels copied from the answer. Correct steps and answers are colored in yellow and incorrect ones in grey.

favors value models over typical reward models. Thus, outcome supervision, leading to value models, is preferred to process supervision that results in reward models, for partial path evaluation.

Outcome supervision is labor-friendly without fine-grained annotations In terms of future orientation, rewards can be modified to introduce such aspects, e.g. "whether steps are correct and *helpful* to the correct answer". We acknowledge that such reward adjustments are useful for planning. However, annotating rewards at the step level is labor-intensive. Furthermore, assessing steps' contribution to final answers, beyond mere correctness, increases the labor demands of reward labeling. In contrast, outcome supervision only requires the final answer's correctness.

4 Experiment Results

4.1 Experimental settings

Benchmarks We conduct experiments on two mathematical reasoning datasets, GSM8K (Cobbe et al., 2021) and Game of 24 (Yao et al., 2023).

Baselines We benchmark our method against leading models in GSM8K and the notable Tree-of-Thought in Game of 24 (Yao et al., 2023), as well as other guided decoding approaches in a relatively fair setting. [3] For comparisons between OVM planning and baseline performance of our generators without guided decoding, see Appendix D.

See the implementation details, including model architecture, training datasets, hyperparameters, and inference algorithm in Appendix A and B.

4.2 Overall Performance

Benchmarking against current state-of-the-art approaches The OVM performance in GSM8K and Game of 24 is detailed in Table 2 and Table 3, respectively. Notably, our Mistral-based 7B model surpasses all models under 70B in GSM8K. In the 7B category, excluding Mistral-based models, our Llama2-based 7B model achieves the highest performance. In the Game of 24, OVM planning significantly improves Llama2-7B's accuracy, increasing its accuracy from 11% to a remarkable 78.7% with 20 sampled solution paths.

[3] There are many hyperparameters in guided decoding, and it is difficult to ensure all are fair.

Table 2: Accuracy on GSM8K. In the third column, we mark models that use GPT for inference or are trained with GPT-generated data. Notably, we don't rely on GPT, data augmentation, and code execution (execute the complete code block outputting the final answer). SC denotes 'Self-Consistency' and RM denotes 'Reward Model'.

Model	Size	GPT-3.54	Data Augmentation	Accuracy
Open-Source Models without Code Execution				
MuggleMATH (Li et al., 2023a)	7B	✓	✓	68.4%
Arithmo-Mistral	7B	✓	✓	74.7%
MetaMath-Mistral (Yu et al., 2023)	7B	✓	✓	77.7%
MetaMath (Yu et al., 2023)	13B	✓	✓	71.0%
MuggleMATH (Li et al., 2023a)	13B	✓	✓	74.0%
RFT (Yuan et al., 2023)	70B			64.8%
WizardMath (Luo et al., 2023)	70B		✓	81.6%
MuggleMATH (Li et al., 2023a)	70B	✓	✓	82.3%
MetaMath (Yu et al., 2023)	70B	✓	✓	84.3%
Ours – OVM (Llama2-7B, K=100)	*7B*			73.7% ± 0.4%
Ours – OVM (Mistral-7B, K=100)	*7B*			**84.7**% ± 0.3%
Open-Source Models with Code Execution				
ToRA-Code (Gou et al., 2023)	7B	✓	✓	72.6%
ToRA-Code (Gou et al., 2023)	13B	✓	✓	75.8%
ToRA-Code (SC, K=50) (Gou et al., 2023)	34B	✓	✓	85.1%
ToRA (Gou et al., 2023)	70B	✓	✓	84.3%
ToRA (SC, K=50) (Gou et al., 2023)	70B	✓	✓	88.3%
Closed-Source Models				
PaLM (SC, K=32) (Huang et al., 2022)	540B			82.1%
DeepMind's+RM Verification (K=96) (Uesato et al., 2022)	70B			87.3%
GPT-4 (Bubeck et al., 2023)	-	✓		87.1%
GPT-4 Code+Self-Verification (K=5) (Zhou et al., 2023)	-	✓		97.0%

Table 3: Accuracy on Game of 24. GPT-4's accuracy is from Yao et al. (2023), and K of ToT is estimated from Figure 3 in their paper.

	Accuracy
GPT-4 CoT	4.0%
GPT-4 SC (K=100)	9.0%
GPT-4 ToT (K=60)	74.0%
Fine-tuned Llama2-7B	11.0%
Fine-tuned Llama2-7B SC (K=100)	11.7% ± 1.3%
Ours – OVM (Llama2-7B, K=20)	78.7% ± 1.7%

Benchmarking against guided decoding approaches Table 4 shows that OVM excells over most approaches listed in guided decoding, except for the GPT-based method. Significantly, OVM surpasses those value-based approaches with a larger model and greater sampling size.

5 Analysis and Discussion

This section seeks to answer the following two **R**esearch **Q**uestions (RQs).

RQ 1. *Can OVM plan?*

RQ 2. *How is outcome supervision compared to process supervision for guided decoding?*

Evaluation with correct answer proportion To assess planning effectiveness, we analyze the proportion of sampled solution paths yielding correct answers in the final sampling stage, immediately preceding the final solution selection. This offers insights into guided decoding's efficiency in steering towards correct answers.

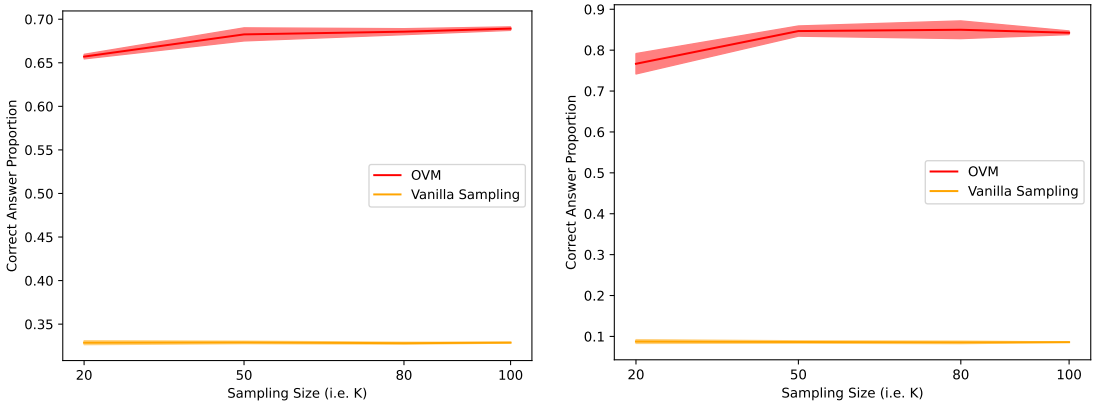
5.1 RQ1: Can OVM plan?

We use 'vanilla sampling" as the baseline for comparison, which relies on random solution sampling based solely on LM probabilities, without guiding.

OVM is an effective planner guiding to the correct answers The result is shown in Figure 3. Notably, in GSM8K, less than 35% of the generator's randomly sampled solution paths are correct, and this proportion increases to over 65% with OVM planning. Similarly, in the Game of 24, OVM planning significantly boosts the correct answer proportion from approximately just 10% in vanilla sampling to an impressive 80%. Additionally, in vanilla sampling, the proportion of correct solutions remains consistent across various sampling sizes. In contrast, OVM planning demonstrates improved benefits with increased sampling sizes, up to a point of saturation.

Table 4: Accuracy on GSM8K comparing with guided decoding approaches. 'RM' denotes 'Reward Model', and 'VM' denotes 'Value Model'.

Model	Size	K	Type	Accuracy
Reward-based				
GRACE-Llama (Khalifa et al., 2023)	7B	20	RM	30.9%
GRACE-T5 (Khalifa et al., 2023)	770M	20	RM	36.3%
Xie et al. (2023) (with GPT3.5-Codex)	175B	80	Prompting	85.5%
Value-based				
RAP (Hao et al., 2023)	33B	10	Simulation	51.6%
CoRe (Khalifa et al., 2023)	12B	40	Simulation	63.2%
Feng et al. (2023)	7B	10	VM	52.2% ± 0.9%
Feng et al. (2023)	7B	10	VM+Voting	56.4% ± 0.6%
Ours – OVM (Llama2-7B)	*7B*	10	VM	66.5% ± 0.2%
Ours – OVM (Mistral-7B)	*7B*	10	VM	**81.2**% ± 0.6%



(a) GSM8K (b) Game of 24

Figure 3: The relationship between the sampling size and correct answer proportion (Llama2-7B)

5.2 RQ2: How is outcome supervision compared to process supervision for guided decoding?

We further compare the performance of reward models, trained under process supervision, with OVM in guiding decoding for GSM8K and Game of 24. The training datasets and hyperparameters for the reward models are identical to those of OVM. Due to resource limitations, we only conducted the experiments on Llama2-7B.

We investigate both typical and modified future-oriented rewards in our study. The former rewards steps (i.e. labeled as 1) for logical correctness; The typical rewards acknowledge steps (labeled as 1) for their logical correctness, while the latter rewards steps that are not only correct but also contribute to the correct final answer. We refer to the model trained with this enhanced reward scheme as PRM-O, denoting its implicit consideration of future Outcomes. See the annotation details in Appendix C.

Comparison in Game of 24 Figure 4 demonstrates the evolving trends across training epochs, illustrating that both OVM and PRM-O effectively guide towards correct answers, in contrast to PRM's failure, highlighting the importance of anticipating outcomes. [4] Notably, PRM-O shows faster convergence compared to OVM, but OVM eventually reaches a performance level comparable to PRM-O. This indicates that outcome supervision, relying only on final answer correctness, may suffice for models to learn outcome evaluation, while more detailed step-level rewards can accelerate this learning process.

Comparison in GSM8K See the results in Table 5. PRM-O outperforms PRM, consistently favoring anticipating outcomes. However, OVM and PRM show similar performance levels. We analyze the reason behind OVM's lack of superior performance over PRM as follows.

[4] Two more training epochs are conducted for better visualization.

Table 5: Correct answer proportion in GSM8K in comparison among OVM, PRM, and PRM-O.

	K=20	K=100
OVM	65.81% ± 0.59%	68.92% ± 0.20%
PRM	65.91% ± 0.66%	69.12% ± 0.28%
PRM-O	67.42% ± 0.60%	70.39% ± 0.15%

Analysis on the difference between Game of 24 and GSM8K Distinct patterns emerge in comparing outcome supervision versus process supervision across Game of 24 and GSM8K. The key lies in the consistency between PRM labels (logical correctness only) and PRM-O labels (also contributes to the correct answer) in Game of 24 and GSM8K, which are 56.9% and 98.6% respectively. This suggests that in Game of 24, logical correctness does not reliably predict answer success. With future-oriented evaluation, value models demonstrate a significant superiority in this scenario. Conversely, in GSM8K, logically correct steps typically yield correct outcomes, thus positioning PRM almost on par with OVM in guiding to the correct answers.

Overall, when considering both the performance and annotation costs, outcome supervision demonstrates superior utility across various settings: it reaches competitive and even better performance than process supervision, with significantly reduced demands on annotation efforts.

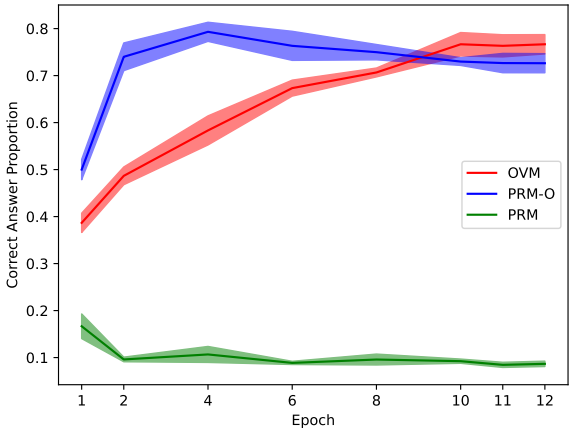


Figure 4: The comparison between OVM, PRM, and PRM-O in correct answer proportion in Game of 24 (K=20)

6 Conclusion

In conclusion, this paper presents a novel approach in verifying intermediate steps and guiding model generation. This is achieved through the introduction of the Outcome-based Value Model (OVM), which employs outcome supervision in training a value model for intermediate steps. Both theoretical and empirical evidence highlight the effectiveness of outcome supervision for value estimation in planning, offer a method that is more efficient and effective than process supervision, which results in a reward-based model. The OVM, requiring no costly step-level annotations and fewer sampled paths, demonstrates superior performance in complex multi-step reasoning tasks, as evidenced by its state-of-the-art results on GSM8K and impressive success rate improvement in the Game of 24.

Limitations

Guided decoding, while effective, introduces a limitation: it introduces an additional model to aid the generator decoding, which imposes a more substantial demand on memory resources and decelerates the inference process. This poses a significant challenge in some real-world applications where rapid response is crucial. Besides, our study does not delve into the costs associated with training a sufficiently accurate value model. While process supervision may enable the training of a reward model with a small dataset, outcome supervision could necessitate a considerably larger dataset for the effective training of a value model. This raises concerns about the scalability of such a system. Additionally, the generalization capability of the value model remains unexplored in our research. This omission leaves unanswered questions regarding the model's adaptability and performance consistency across diverse or unforeseen scenarios.

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The*

Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*, abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *CoRR*, abs/2309.17179.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *CoRR*, abs/2309.17452.

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. *CoRR*, abs/2305.14992.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *CoRR*, abs/2210.11610.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Discriminator-guided multi-step reasoning with language models. *CoRR*, abs/2305.14934.

Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. 2023a. Query and response augmentation cannot help out-of-domain math reasoning generalization. *CoRR*, abs/2310.05506.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *CoRR*, abs/2206.02336.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5315–5333. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *CoRR*, abs/2305.20050.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR*, abs/2308.09583.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *CoRR*, abs/2210.03350.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Decomposition enhances reasoning via self-evaluation guided decoding. *CoRR*, abs/2305.00633.

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. *CoRR*, abs/2305.10601.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *CoRR*, abs/2309.12284.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. *CoRR*, abs/2308.07921.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2023. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4471–4485. Association for Computational Linguistics.

A Training Method

Model architecture The model is a decoder-only autoregressive language model. Following Cobbe et al. (2021), we add a linear layer with a single bias parameter after the model's final unembedding layer.

Training set Given a training question q, the generator is queried to generate n solution paths $\mathbb{S} = \{S_1, \cdots, S_n\}$. Therefore, with a set of N questions $\mathcal{Q}$, we can obtain $N \times n$ question-path pairs. The training labels are based on either outcome supervision or process supervision, where the former can get the labels automatically via rules like string matching to the ground answer while the latter one is more labor-intensive.

Outcome supervision The label is dependent on the final answer a, which is binary $y_o \in \{0, 1\}$, indicating whether a is correct or not. Since it is often trained with token-level supervision, the label is expanded into a consistent vector $\boldsymbol{y}^o = [y_o, \ldots, y_o]$ of token sequence length.

Process supervision The label is a vector with each dimension indicting whether the corresponding step is correct or not, i.e. $[y_1, \ldots, y_m]$, where m is the number of steps. After expanding into token level, the label becomes $\boldsymbol{y}^p = [y_1, \ldots, y_1, \ldots, y_m, \ldots, y_m]$, which assigns the same label y_i to all tokens within the i-th step.

Training objective The objective is a mean squared error loss,

$$L(S, \boldsymbol{y}^l; q) = \left(f(q; S) - \boldsymbol{y}^l\right)^2, \quad l = o \text{ or } p$$

Additionally, the model is jointly trained with language modeling loss unweighted, following Cobbe et al. (2021).

B Implementation of Guided Decoding with OVM

Building training dataset for OVM Given the training question set, we first sample 100 solution paths for each question from the generator, with a temperature setting to 0.7. Then, we detect the answer correctness for each sample. In GSM8K, the answer correctness is determined by exact string matching to the ground truth since all the answers are integers. In Game of 24, the answer correctness is based on the validity of the equation equating

Algorithm 1 Value-Guided Beam Search

Input: Question q, Beam size b, Sampled steps per state K, Maximum step count T
Output: Best solution sequence for q
Model: Generator Φ and OVM f
procedure VALUEGUIDEDBEAMSEARCH(q, b, K)
  Initialize step sequences $\mathbb{S} \leftarrow \{\}$
  Sample initial steps $\{s_1^1, \ldots, s_K^1\}$
  Evaluate values $\{v_1^1, \cdots, v_K^1\}$ for each step
  Select top b valued steps and add to $\mathbb{S}$
  $t \leftarrow 1$
  while sequences in $\mathbb{S}$ are not complete and $t < T$ **do**
    $\mathbb{S}_{\text{new}} \leftarrow \{\}$
    $\mathcal{V} \leftarrow \{\}$
    for each sequence $S^{(1:t)}$ in $\mathbb{S}$ **do**
      for $i = 1$ to K/b **do**
        $S_i^{(1:t+1)} = \Phi(S^{(1:t)}; q)$
        $v_i^{(1:t+1)} = f(S_i^{(1:t+1)}; q)$
        $\mathbb{S}_{\text{new}} \leftarrow \mathbb{S}_{\text{new}} + S_i^{(1:t+1)}$
        $\mathcal{V} \leftarrow \mathcal{V} + v_i^{(1:t+1)}$
      end for
    end for
    $\mathbb{S}_{\text{new}} \leftarrow$ top b valued sequences from $(\mathbb{S}_{\text{new}}, \mathcal{V})$
    $\mathbb{S} \leftarrow \mathbb{S}_{\text{new}}$
    $t \leftarrow t + 1$
  end while
  return sequence with highest final value in $\mathbb{S}$
end procedure

to 24 and the singular usage of input numbers, following (Yao et al., 2023). This allows us to produce numerous OVM training samples using just question-answer pairs, without path annotations, resulting in 747,300 for GSM8K and 90,000 for Game of 24.

Inference - beam search with guided decoding We conduct the beam search strategy guided by OVM during inference. Diverging from the traditional token-level probability-guided beam search, it is guided by the step-level estimated values. The algorithm is detailed in Algorithm 1. We set the maximum number of steps as 10 and the maximum new token length is 400. We report the best results across available beam sizes, e.g. the result of $K = 20$ is the best one among $b = 2, b = 4, b = 5, b = 10$. Each inference experiment was repeated 3 times, with the average results and their standard variation reported.

Training generators We use the newline character as the marker for the end of each step. In GSM8K, we fine-tuned Mistral-7B and Llama2-7B on the training set. Given that GSM8K provides calculation annotations, our models were also trained to utilize calculators. In Game of 24, we fine-tuned Llama2-7B on problems indices 1-900 with enumerated solution paths. For both datasets, the fine-tuning was carried out for 2 epochs, with a batch size of 128. We set the maximum learning rate to 1e-5, using a linear scheduler with the AdamW optimizer. We implement FlashAttention for Llama2-7B (Dao et al., 2022, Dao, 2023).

Training hyperparameters for OVMs/ORMs [5] OVMs were initialized from the corresponding generator checkpoints. In GSM8K, OVM was trained for 1 epoch with a batch size of 512. In Game of 24, OVM is trained for 10 epochs with a batch size of 128, due to its smaller training set. The optimizer was AdamW and the maximum learning rate was set to 1e-5 for Llama2-7B and 2e-6 for Mistral-7B respectively, following a linear scheduler.

C Process Label Annotation and Consistency Evaluation

In Game of 24, process labels are derived by checking the syntax and calculation, and matching to all possible correct solutions, enumerated by rules. For PRM training, steps are labeled as "correct" when the steps are logically correct in rules, i.e. only use each given number once to calculate correctly. For PRM-O training, we match steps to the enumerated possible correct solutions, and they are labeled as "correct" only when the matching succeeds.

In GSM8K, we query GPT-4 to annotate process labels without references. GPT-4 is asked to classify each step into "correct", "incorrect", or "unnecessary". The used prompt is shown as follows:

[Question]
`question`

[Correct Answer]
`answer`

[Solution]
`solution path`

[System]
You are an expert math examiner. Review the student's solution and mark each step as correct only if it's based on accurate premises and helps solve the

[5] Since we train the value model with outcome supervision, the objective originally intended for ORM, the same model is simultaneously used as OVM for partial paths and ORM for complete paths in this paper.

problem. Mark it as "unnecessary" when it is logically valid but doesn't help. Please mark with "[Conclusion]" and summary all your judgements in the format of "Step i is correct/incorrect/unnecessary".

For PRM training, "unnecessary" steps are converted to "correct", while for PRM-O training, they are converted to "incorrect".

We conduct a consistency evaluation of GPT-4 labeling compared to human labeling on a small set. To ensure coverage of this set across paths of different lengths, we randomly select two solutions from each length set, including one with the correct final answer and one with an incorrect answer. For instance, we sample a correct solution and an incorrect solution from both the 1-step path set and the 2-step path set. Additionally, we apply the same sampling procedure to the sets classified by the step length of reference solutions. Finally, we get 116 question-solution pairs. We hire three master-level students to annotate those examples as "correct", "incorrect", or "unnecessary". The Pearson correlations on these examples labeled by GPT-4 and humans are shown in Table. This indicates GPT-4 can provide process labels with high consistency to humans.

	Human 1	Human 2	Human 3	GPT-4
Human 1	-	0.89	0.88	0.87
Human 2	0.89	-	0.91	0.86
Human 3	0.88	0.91	-	0.86
GPT-4	0.87	0.86	0.86	-

Table 6: Pearson correlations on 116 samples between GPT-4 labeling and human labeling

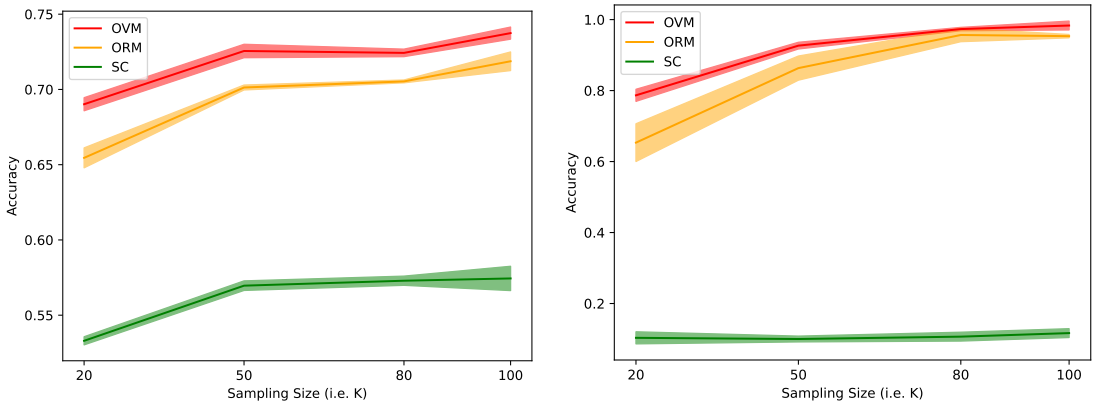
D More Results on OVM Planning

We compare the effectiveness of OVM planning with the non-planning version of the same model we implemented, including greedy, Self-Consistency (SC), and verification by Outcome Reward Model (ORM). The latter two are approaches for post-processing multiple solutions by vanilla sampling without guided decoding.

Table 7 compares our OVM planning with non-planning implementations. The findings indicate that OVM, when used for guided decoding, typically surpasses post-selection with ORM, provided the number of sampled paths is the same. However, an exception occurs with Mistral-7B at K=100, where the gap between OVM and ORM approaches appears to reach saturation. As shown in Figure 5, in both GSM8K and Game of 24, the accuracy improves as the sampling size increases. The gap between ORM and OVM performance tends to decrease as there are more sampled paths.

Table 7: Accuracy on GSM8K and Game of 24. Average and standard variation of 3 runs are reported. K denotes sampling size.

Method			GSM8K		Game of 24
			Llama2-7B	Mistral-7B	Llama2-7B
w/o Guided Decoding	Greedy		38.6%	58.4%	11%
	SC	K=20	53.3% ± 0.3%	70.2% ± 0.7%	10.3% ± 1.7%
		K=100	57.4% ± 0.8%	72.6% ± 0.2%	11.7% ± 1.3%
	ORM	K=20	65.5% ± 0.7%	81.8% ± 0.2%	65.3% ± 5.3%
		K=100	71.9% ± 0.6%	**84.7**% ± 0.4%	95.3% ± 0.5%
Planning	OVM	K=20	69.0% ± 0.4%	82.6% ± 0.1%	78.7% ± 1.7%
		K=100	73.7% ± 0.4%	**84.7**% ± 0.3%	**98.3**% ± 1.2%



(a) GSM8K (b) Game of 24

Figure 5: The relationship between the sampling size and accuracy (Llama2-7B)

	Complete path evaluation	Partial path evaluation
Process supervision	Reward	Reward
Outcome supervision	Reward	Value

Table 8: Types of scores predicted by process- or outcome-supervised models on the complete path and partial path, respectively. When evaluating partial paths, the predicted scores of outcome-supervised models are values.