

Your Language Model May Think Too Rigidly: Achieving Reasoning Consistency with Symmetry-Enhanced Training

Yihang Yao¹, Zhepeng Cen¹, Miao Li¹, William Han¹, Yuyou Zhang¹, Emerson Liu², Zuxin Liu³, Chuang Gan⁴, Ding Zhao¹

¹Carnegie Mellon University, ²AHN, ³Salesforce AI Research, ⁴UMass Amherst
{yihangya, zcen}@andrew.cmu.edu

Abstract

Large Language Models (LLMs) have demonstrated strong reasoning capabilities across various tasks. However, even minor variations in query phrasing, despite preserving the underlying semantic meaning, can significantly affect their performance. To address this, we focus on enhancing LLMs’ awareness of symmetry in query variations and propose syMmetry-ENhanceD (MEND) Data Augmentation, a data-centric approach that improves the model’s ability to extract useful information from context. Unlike existing methods that emphasize reasoning chain augmentation, our approach improves model robustness at the knowledge extraction stage through query augmentation, enabling more data-efficient training and stronger generalization to Out-of-Distribution (OOD) settings. Extensive experiments on both logical and arithmetic reasoning tasks show that MEND enhances reasoning performance across diverse query variations, providing new insights into improving LLM robustness through structured dataset curation.

1 Introduction

Large Language Models (LLMs) have demonstrated superior performance across various reasoning tasks, including mathematical reasoning (Ye et al., 2024a; Qu et al., 2024; Lin et al., 2024a; Gou et al., 2023; Shen et al., 2025; Cen et al., 2025), code generation (Chen et al., 2021; Zhang et al., 2024, 2023; Dainese et al., 2024), and autonomous system decision-making (Yang et al., 2024b; Sima et al., 2024; Li et al., 2025). Despite their ability to handle complex reasoning tasks, LLMs exhibit naive failure modes. For instance, they suffer from premise order sensitivity (Chen et al., 2024; Zhu et al., 2024a), the reverse-curse phenomenon (Berglund et al., 2023; Golovneva et al., 2024), and distractibility (Shi et al., 2023a; Zhu et al., 2024b), making them vulnerable to variations in the natural language description of a query,

even when the underlying semantic meaning remains unchanged.

A reasoning problem can be structurally decomposed into two layers (Zhu et al., 2023; Zhou et al., 2024): the underlying layer represents the *semantic meaning*, which includes the structure of reasoning chains and the knowledge required for logical deduction, while the upper layer consists of the *surface form*, referring to the natural language description of the problem (Zhou et al., 2024). The inability of LLMs to maintain consistent performance across different surface forms highlights the need for *reasoning consistency*, which we define as the model’s ability to generate consistent and correct answers from different query transformations of a problem with the same semantic meaning. In this work, we explore the question: *how to improve LLMs’ reasoning consistency across varying surface forms*.

Our key insight is that LLM reasoning problems maintain certain underlying structures, such as *description symmetry*, which describes transformations of the query description that preserve its semantic meaning. Existing LLM post-training methods typically focus on improving the quality of training data for reasoning chains, such as using bootstrapping to enhance reasoning diversity (Yu et al., 2023). However, these methods overlook the description symmetry in reasoning tasks, which naturally exists in surface form variations. As a result, they often suffer from overfitting and inconsistent reasoning with respect to different surface forms as shown in Figure 1. To address this issue, we propose syMmetry-ENhanceD (MEND) Data Augmentation for LLM post-training. Our method augments the dataset with the symmetry information, enforcing the model’s capability to better capture semantic meanings in different query variations, thereby improving post-training data efficiency and enhancing OOD generalization. The main contributions of this work are:

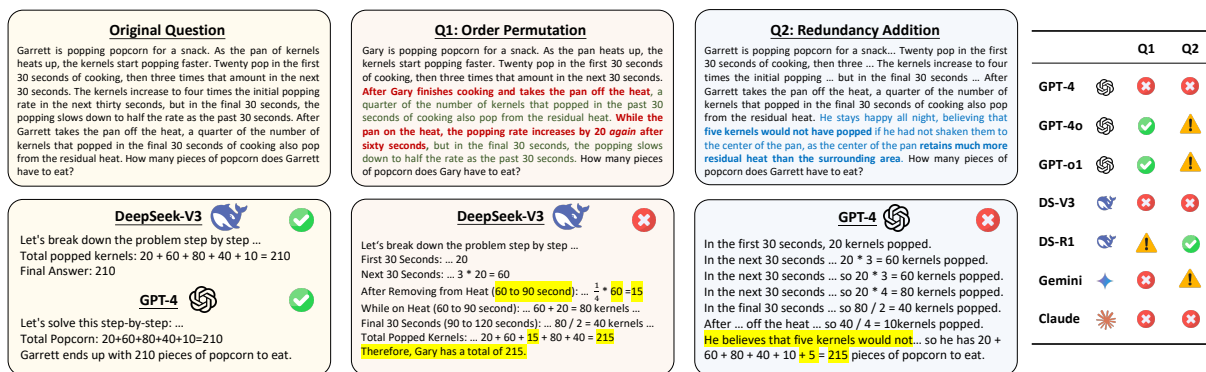


Figure 1: Failure examples of LLMs under surface form variations. Queries are modified from R-GSM (Chen et al., 2024). Table: The correctness for 10 evaluations across different LLMs. ✔: all correct; ✘: all wrong; ⚠: error occurs. Full incorrect answers are provided in Appendix A.5.

1. Formal analysis of the reasoning consistency problem. We associate reasoning consistency in terms of an LLM’s ability to extract invariant knowledge, providing a formal framework for improving reasoning consistency.

2. Introduction of MEND to address the consistency of reasoning through post-training. To our knowledge, this is the first systematic work that investigates post-training techniques to mitigate issues of consistency of reasoning.

3. Extensive evaluation of MEND in LLM reasoning tasks. Our experimental results on PromptBench (Zhu et al., 2023) and iGSM (Ye et al., 2024b) show that MEND achieves superior sampling efficiency and generalizability in reasoning consistency with a significantly enhanced in-context knowledge extraction capability.

2 Related Works

LLM Reasoning: Recent studies highlight the remarkable reasoning capabilities of LLMs, exploring in-context learning (Wei et al., 2022; Yao et al., 2022), pre-training (Ye et al., 2024a,c; Shao et al., 2024; Lightman et al., 2023; Lin et al., 2024b), and post-training (Ni et al., 2024). A key focus in training LLMs is the curation of high-quality datasets for instruction tuning (Yue et al., 2023; Liu et al., 2024; Ni et al., 2024). Another approach involves leveraging LLM-generated datasets (Wang et al., 2024a; Cen et al., 2024), often with reinforcement learning (Ouyang et al., 2022; Kumar et al., 2024). Both strategies highlight the essential role of high-quality data in enhancing LLM reasoning performance.

LLM Failure Modes: Despite their advanced reasoning capabilities, LLMs exhibit surprising brittleness to variations in question descriptions with

the same semantical meaning (Chen et al., 2024). For example, the *Reversal Curse* (Berglund et al., 2023) refers to LLMs failing to generalize from statements like “A is B” to “B is A.” *Premise Ordering* describes the performance degradation when the order of premises in a query differs from the order in their reasoning chains (Chen et al., 2024). *Distractibility* is another failure mode in which LLMs’ reasoning performance declines when irrelevant context is included in the query (Shi et al., 2023b). To mitigate these issues, researchers have explored inference-time scaling methods that paraphrase queries (Zhou et al., 2024) and post-training techniques (Golovneva et al., 2024) that reverses the order of tokens in training. However, a systematic analysis and solution to address these robustness challenges still remain an open problem.

Symmetry and Equivariant Learning: Symmetry has been widely used to indicate rotationally symmetric problems, such as image-input machine learning tasks (Weiler and Cesa, 2019). The definition of symmetry can also be extended to groups that preserve structured information while performing transformations (Muglich et al., 2022). Encoding data symmetries in the model training pipeline can improve both generalization and sample efficiency (Wang et al., 2022), an idea first proposed in G-Convolution (Cohen and Welling, 2016). Recent works also incorporate equivariant learning in reinforcement learning (Liu et al., 2023; Wang et al., 2022) and robotics (Yang et al., 2024a). The symmetry in the training corpus of LLMs remains a widely unexplored area.

3 Preliminary for LLM Reasoning

DAG Representation. Following previous works (Zhu et al., 2023, 2024a; Ye et al., 2024b),

we formulate the reasoning task as a problem defined over a directed acyclic graph (DAG) representation: $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ represents the set of nodes, and $E \subseteq V \times V$ represents the set of directed edges indicating dependencies or relationships between nodes. The root node $v_r \in V$ corresponds to the target variable we aim to compute or reason about, leaf nodes $v_l \in \mathcal{L} \subseteq V$ denote the variables with known values, and the other nodes $v_i \in \mathcal{I} = V \setminus (\mathcal{L} \cup \{v_r\})$ represent intermediate nodes that need to be computed by their parent nodes, which are denoted as $\text{Pa}(v_i) = \{v_j \mid (v_j, v_i) \in E\}$. Each directed edge $(v_j, v_i) \in E$ indicates that the value of x_i depends on x_j and their quantitative relationship.

Ground Truth of Reasoning Chain: The ground truth reasoning path for the reasoning chain is represented as a sequence of intermediate reasoning steps $\{y_1, y_2, \dots, y_T, y\}$ to the final result y , where y_t corresponds to the result of the t -th intermediate computation. This process is grounded in the structure of the directed acyclic graph $G = (V, E)$. The reasoning process follows a topological sorting of G , which is a linear ordering of its nodes such that for every directed edge $(v_i, v_j) \in E$, node v_i appears before v_j in the ordering.

4 MEND Data Augmentation

In this section, we first analyze the surface form and its transformations and reveal the symmetry in description queries. We then discuss LLMs' capability for reasoning consistency. Lastly, we introduce the data augmentation method to improve reasoning consistency and a probing tool for capability verification.

4.1 Surface Form and Description Symmetry

A reasoning problem consists of both a semantic meaning and a surface form (Zhou et al., 2024). The semantic meaning is determined by the structure of the DAG and the values of its leaf nodes, capturing the core reasoning chain of the problem (Pearl, 2009; Velickovic and et al., 2021). The surface form is the natural language description of the semantic meaning (Zhou et al., 2024). There exists a mapping $f : \mathcal{Q} \rightarrow \mathcal{S}$ from the query space \mathcal{Q} to the semantic space \mathcal{S} , which translates a natural language description into its underlying structured representation. This abstracts away linguistic or surface-level variations to focus on the task's core logical or mathematical structure (Russell and

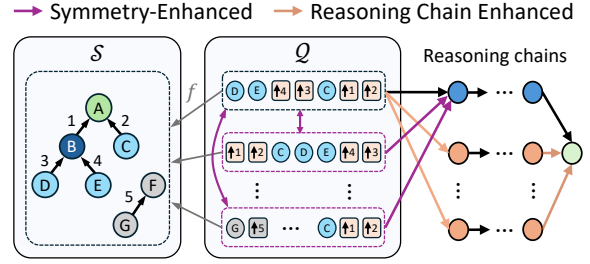


Figure 2: Overview of Symmetry-Enhanced Data Augmentation and its Comparison with Reasoning Chain Data Augmentation.

Norvig, 2021).

The symmetry relationship \sim on \mathcal{Q} is defined as:

$$q_1 \sim q_2 \iff f(q_1) = f(q_2), \quad (1)$$

where $q_1, q_2 \in \mathcal{Q}$. That is, two query descriptions are symmetric if they correspond to the same semantics in \mathcal{S} . We categorize symmetry relations in \mathcal{Q} into three types: **(1) Permutation:** reordering the provided information, **(2) Redundancy addition:** introducing irrelevant information as a distraction, and **(3) Surface-level variations:** paraphrasing questions at the linguistic level. Given the complexity of linguistic paraphrasing, this work primarily focuses on the first two types, leaving the third for future research.

4.2 LLM Reasoning Consistency

Given a query q , the reasoning process to find its ground-truth final answer y of an auto-regressive LLM can be formulated as a conditioned generation process (Xiang et al., 2025):

$$\mathcal{P}(y|q) = \int_z \prod_{t=1}^T \mathcal{P}(y_t | y_{<t}, z, q) \mathcal{P}(z|q) dz, \quad (2)$$

where $\mathcal{P}(y|q)$ means the probability of generating right final answer and $\mathcal{P}(z|q)$ represents **in-context knowledge extraction**, which extracts relevant knowledge z from q . $\mathcal{P}(y_t | y_{<t}, z, q)$ represents the conditioned generation to the final results step by step.

We define the **reasoning consistency** as the ability of an LLM to maintain similar reasoning performance despite variations in the phrasing or description of a query q , as long as its underlying semantics remains unchanged, i.e.,

$$\mathcal{P}(y | q_1) \approx \mathcal{P}(y | q_2), \quad q_1 \sim q_2 \quad (3)$$

Equation 2 demonstrates that extracting high-quality representations z from queries is crucial for

ensuring reasoning consistency. In this work, we focus on *invariant knowledge extraction*: improving the LLMs’ capability to consistently extract useful z despite the surface form changes.

4.3 Symmetry-Enhanced Data Augmentation

To improve representation extraction capability, we can adopt either *model-centric* approaches, which focus on modifying model architectures, or *data-centric* approaches, which aim to improve data quality. Given the complexity of modifying the architecture of pre-trained LLMs without compromising overall performance, we focus on a data-centric approach that encodes symmetry information in the post-training dataset and is compatible with most general LLM architectures. We propose syMmetry-ENhanced (MEND) Data Augmentation as shown in Figure 2. Compared to previous methods that enhance the training chains in the training dataset (Yu et al., 2023), we focus on query augmentation to impose a better understanding of symmetric structured information. Our method contains two parts:

Step 1: Order Permutation. We begin by splitting the original query q into a list of partitions, using the newline character (“\n”) as the delimiter. Next, we shuffle the order of these partitions while preserving the semantic meaning, resulting in a reordered list of sentences.

Step 2: Redundancy Addition. To add redundant information to the original question, we first randomly sample some new nodes and edges, which are not connected to the original DAG so that they will not contribute to the computation of the target node, and then generate the new premises accordingly following the previous template. After this, we add the partition to a random position of the list, and then combine them together forming a query q' in the augmented dataset \mathcal{D}_{aug} .

More details are presented in Alg. 1. After the data augmentation, we finetune the language model using supervised fine-tuning (SFT) on the augmented dataset \mathcal{D}_{aug} .

4.4 Knowledge Extraction Verification Tool

In the previous subsections, we proposed MEND to enhance knowledge extraction capability and address reasoning consistency requirements. However, it remains unclear whether MEND fundamentally improves reasoning capability or merely relies on memorization. In this section, we introduce the probing tool we use to evaluate the representation

Algorithm 1 MEND

Input: QA pair (q, a) from the original dataset, augmentation times K redundant information number R , separation delimiter SEP.

Output: Augmented dataset \mathcal{D}_{aug} for this QA pair

```

1: Initialize  $\mathcal{D}_{\text{aug}} \leftarrow \{(q, a)\}$ 
2: for  $i = 1, \dots, K$  do
3:   # Step 1: Order Permutation
4:   Divide the query into a segmentation list  $L$  by SEP;
5:   random.shuffle( $L$ )
6:   # Step 2: Redundancy Addition
7:   for  $j = 1, \dots, R$  do
8:     Randomly sample redundant nodes with random values and edges;
9:     Construct a redundant partition  $l_r$  by applying template to the new nodes;
10:     $L.append(l_r)$ ;
11:   end for
12:   random.shuffle( $L$ );
13:    $q' \leftarrow \text{SEP.join}(L)$ 
14:    $\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{aug}} \cup \{(q', a)\}$ 
15: end for
16: Return:  $\mathcal{D}_{\text{aug}}$ 

```

extraction capability, providing an explanation for the performance enhancement brought by MEND.

Recent literature has revealed that in transformer-based LLMs, attention patterns are good indicators of whether LLMs can effectively retrieve useful information from queries (Wang et al., 2024b; Hou et al., 2023). Based on these findings, we utilize the attention of LLMs to analyze the knowledge extraction capability.

The core component of the attention-based probing is a binary classification task to determine whether a statement $q[i]$ in question query q contains any useful information (Hou et al., 2023):

$$\mathcal{P}(q[i]|\mathbf{A}) \rightarrow \{0, 1\}, \quad (4)$$

where $\mathbf{A} = \{\mathbf{A}(l, h) \mid 1 \leq l \leq L; 1 \leq h \leq H\}$ is the combination of attention weights across L layers and H heads after processing the question query q . The prediction result 1 indicates $q[i]$ is useful in reasoning, and 0 indicates $q[i]$ only contains irrelevant information. Following the previous work (Hou et al., 2023), we segment all tokens in a query into multiple groups and construct simplified attention \mathbf{A}_{simp} by pooling in each group as the conditions to reduce the dimension of the

attention weights:

$$\mathcal{P}(q[i]|q, \mathbf{A}) \approx \mathcal{P}(q[i]|q, \mathbf{A}_{\text{simp}}), \quad (5)$$

The previous attention-based probing method (Hou et al., 2023) employs non-parametric algorithms, such as KNN, for query information retrieval. However, it struggles to accurately identify relevant premises, particularly in complex tasks or larger networks. This limitation arises because KNN treats all attention entries equally, overlooking the fact that the information at each position is inherently influenced by preceding content. As input length and network depth increase, this aggregation effect becomes more pronounced, further undermining the probing method’s effectiveness. To address this issue, we adopt a linear probing approach based on logistic regression:

$$\mathcal{P}(q[i]|\mathbf{A}_{\text{simp}}) = \sigma(\mathbf{w}^\top \mathbf{A}_{\text{simp}} + b), \quad (6)$$

where \mathbf{w} and b are trainable variables. See more discussions on comparison between KNN-based and our probing methods in Appendix A.4.

5 Experiments

In the experiments, we are going to answer the following research questions. **(RQ-1)** How do current LLMs perform in terms of reasoning consistency? **(RQ-2)** How does MEND benefit reasoning consistency? **(RQ-3)** Why does MEND improve reasoning consistency? To answer these questions, we make the following experiment setup.

5.1 Experiment setup

Tasks: We evaluate our approach on two benchmarks, PromptBench (Zhu et al., 2023) and iGSM (Ye et al., 2024b). For PromptBench (Zhu et al., 2023), we use two categories of tasks, logical reasoning and arithmetic reasoning. The iGSM assesses LLMs’ math and common-sense reasoning capabilities. The questions from PromptBench can be represented as tree structures, where each non-leaf node has up to two parent nodes. The objective is to compute the value of the root node given the values of the leaf nodes and the computational rules from parent nodes to child nodes. The iGSM tasks can be represented as the more general Graph structure. Two example QA pairs are provided in Appendix B.3.

In logical reasoning tasks, each node takes a boolean value, and the computational rules are

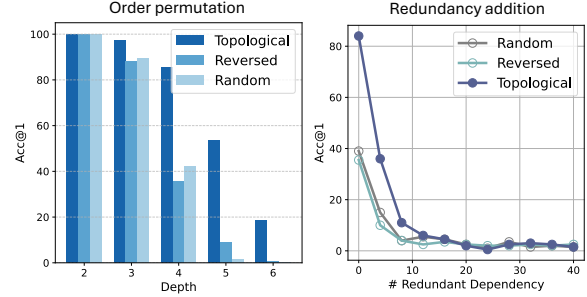


Figure 3: Accuracy evaluation of the DeepSeek-math-7B-base model on the arithmetic reasoning task with different surface forms. Left: results with different DAG depth; Right: results with different redundant information addition.

sampled from $\{\wedge(\text{AND}), \vee(\text{OR}), \neg(\text{NOT})\}$. In arithmetic reasoning tasks, each node takes an integer value, with leaf node values sampled from $0 \sim 10$. The computational rules are drawn from $\{+, -, \times, \square^2\}$. Given an underlying tree representation, PromptBench first assigns a random name to each node and then generates a natural language query using a fixed template, as shown in Appendix B.2.1.

For the evaluation dataset, we create variations of surface forms with both order permutation and redundancy addition. For order permutation, we choose Topological: the topological sorting of the underlying DAG, which aligns with the reasoning chain (Zhu et al., 2023), Random: random permutation of sentences in the question queries, and Reversed: the inversed order of Topological. We utilize the templates in the benchmark (Zhu et al., 2023) to ensure that the overall semantical meaning does not change with the order permutation. For redundancy addition, we add irrelevant descriptions in the query and set the number of redundant dependencies ranging from 0 to 40. We report the zero-shot and pass@1 accuracy unless specified otherwise.

Models: For all experiments, we utilize the Llama-3.2 1B and 3B models (Dubey et al., 2024) as our base models unless specified otherwise.

Baselines: We compare our method with three types of baselines: **(1) Post-training:** Vanilla: trains base models on the vanilla dataset using SFT; and RC-Aug: utilizes the reasoning chain (RC) augmented dataset for SFT (Yu et al., 2023). Specifically, we augment the answers with different topological orderings while keeping the query unchanged; **(2) Inference scaling:** SCoP- k (Zhou et al., 2024), first paraphrases the question prompts

Table 1: Accuracy (%) evaluation on arithmetic reasoning datasets with different permutation order and base model as Llama-3.2-3B. The difficulty level is defined by the number of reasoning steps for ground-truth reasoning chains. The number in the parentheses indicates the performance comparison with the Vanilla method. **Bold**: the method with the best performance.

Order	Method	Difficulty Level				Avg.
		1	2	3	4	
Topological	Vanilla	100.0	99.0	79.5	32.0	77.6
	RC-Aug	100.0	98.5	87.0	33.0	79.6 (+2.0)
	SCoP-2	100.0	82.5	16.0	1.5	50.0 (-27.6)
	SCoP-4	100.0	94.5	28.5	3.5	56.6 (-21.0)
	SCoP-8	100.0	100.0	49.5	9.0	64.6 (-13.0)
	MEND	97.5	99.5	88.5	37.5	80.8 (+3.2)
Random	Vanilla	99.0	65.5	10.5	0.0	43.8
	RC-Aug	100.0	62.5	13.0	1.0	44.1 (+0.3)
	SCoP-2	99.0	77.5	15.0	3.0	48.6 (+4.8)
	SCoP-4	100.0	96.5	26.0	4.5	56.8 (+13.0)
	SCoP-8	100.0	99.5	38.5	12.5	62.6 (+18.8)
	MEND	97.5	99.5	85.5	37.5	80.0 (+36.2)
Reversed	Vanilla	99.5	45.0	5.5	1.0	37.8
	RC-Aug	100.0	30.5	0.5	0.5	32.9 (-4.9)
	SCoP-2	99.5	85.0	20.5	2.5	51.9 (+14.1)
	SCoP-4	100.0	97.0	30.0	4.5	57.9 (+20.1)
	SCoP-8	100.0	100.0	42.5	8.0	62.6 (+24.8)
	MEND	97.5	98.5	86.0	41.0	80.8 (+43.0)

k times, then performs reasoning based on these paraphrases, and obtains the final answer via majority voting. It is combined with vanilla models unless specified otherwise. **(3) Ablation:** In addition to these baselines, we also create a variant of our method called MEND-RC that uses MEND to transform the queries, while augmenting the dataset with more reasoning chains. We evaluate the models with greedy generation (temperature=0) and report the averaged accuracy on 200 testing samples on every evaluation dataset unless specified otherwise.

5.2 Reasoning-Consistency Evaluation

To answer **RQ1**, we evaluate several LLMs with query variations and present the reasoning accuracy in Figure 3. Due to space limitations, we defer the additional results to Appendix A.1. We observe two key findings: (1) The topological order improves the accuracy of generation, while adversarial permutations, including random and reversed orders, degrade performance. Furthermore, the harder the reasoning task (i.e., greater DAG depth), the more pronounced the performance drop, consistent with previous findings for closed-source LLMs (Chen et al., 2024). (2) Redundant information in the question query negatively affects reasoning performance. As the amount of irrelevant information in-

Table 2: Accuracy (%) evaluation on iGSM datasets with different permutation order and base model as Llama-3.2-1B. The difficulty level is defined by the number of reasoning steps for ground-truth reasoning chains. The number in the parentheses indicates the performance comparison with the Vanilla method. **Bold**: the method with the best performance.

# operation		5	10	15	20	Avg.
Random	Vanilla	97.5	85.0	58.5	43.0	71.0
	SCoP-2	97.0	84.5	63.5	47.5	73.1 (+1.9)
	SCoP-4	96.5	90.0	67.5	52.5	76.6 (+5.6)
	MEND	100.0	94.5	77.5	59.0	82.8 (+11.8)
Reversed	Vanilla	95.0	85.0	55.5	53.0	72.1
	SCoP-2	96.5	87.0	57.5	55.0	74.0 (+1.9)
	SCoP-4	97.5	88.5	68.0	64.5	79.6 (+7.5)
	MEND	98.0	94.0	76.0	69.5	84.4 (+12.3)
Topological	Vanilla	96.0	84.5	59.5	52.0	73.0
	SCoP-2	97.0	85.5	63.0	54.5	75.0 (+2.0)
	SCoP-4	97.5	89.5	68.0	60.5	78.9 (+6.9)
	MEND	98.5	94.0	79.0	65.0	84.1 (+11.1)

creases, the model’s reasoning accuracy decreases.

Both observations indicate that open-source models still suffer from **overfitting to topological order and redundancy-free queries**, leading to poor performance in reasoning consistency tasks. This issue further motivates the development of techniques to enhance reasoning-consistency robustness in LLMs.

5.3 Reasoning-Consistency Enhancement

To answer **RQ2**, we present the main experiments with two main parts: reasoning consistency on order permutation and redundancy addition.

Order permutation: For the PromptBench task, we train the LLM on arithmetic questions with difficulty level 2 and 3 while testing its performance on difficulty level 1 ~ 4 with three different permutation orders: Topological, Random and Reversed. The evaluation results on Llama-3.2-3B are listed in Table 1. For the iGSM task, we fine-tune a LLaMA-3.2-3B model on the iGSM training set, which includes samples with operation numbers from 10 to 12. For evaluation, we use a test set covering a broader range of operation numbers (from 10 to 20, larger operation number indicates higher difficulty) and three distinct permutation orders: Topological, Random, and Reversed. The results are presented in Table 2. More results with different base models are available in the Appendix A.2.

Table 1 shows that training solely on the topological order corpus results in overfitting - it achieves

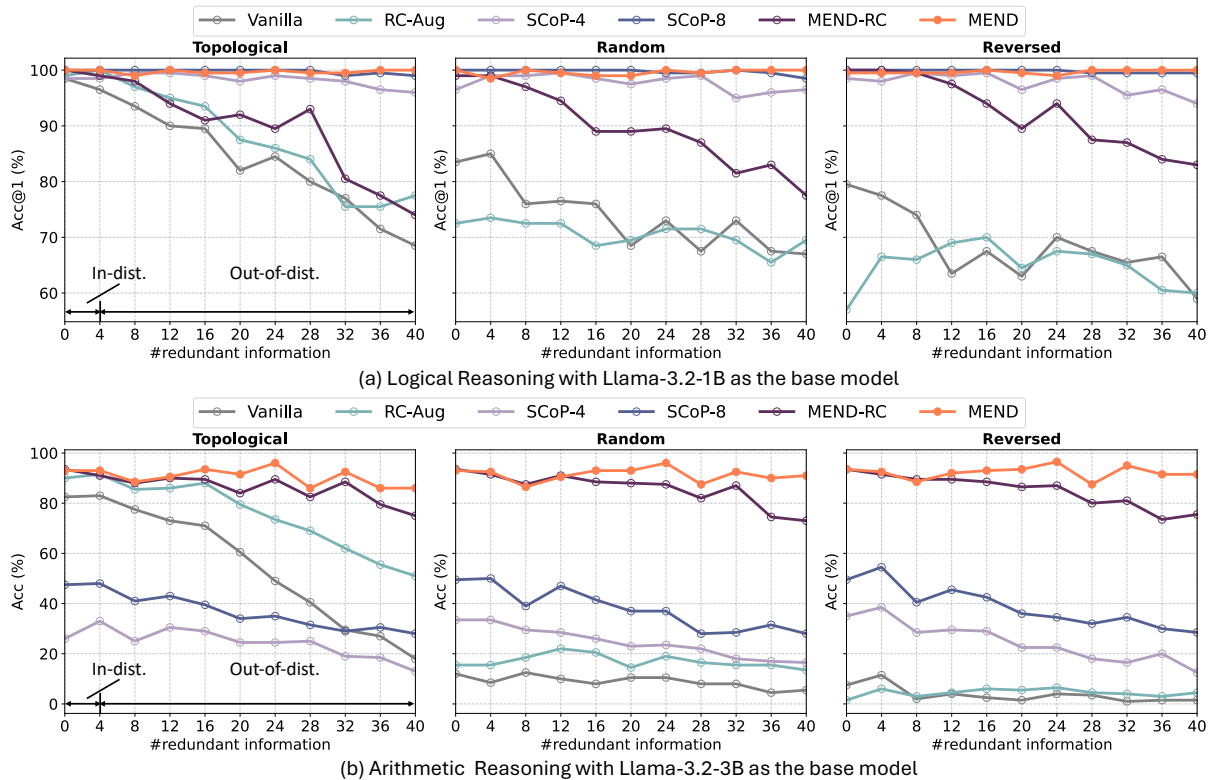


Figure 4: Evaluations with respect to different query variations. Each figure refers to one permutation order type, the x-axis represents the number of redundancies of the test set, and the y-axis represents the accuracy of final answers. For each dataset, we report the accuracy value over a dataset with a size of 200.

superior performance on the Topological test set but suffers significant degradation on the Random and Reversed sets. RC-Aug provides only minor improvements in overall performance. SCoP- k , the inference-time scaling baseline, improves performances on the Reversed and Random sets but degrades on the Topological set. In contrast, our method demonstrates consistently strong performances on all evaluation datasets, effectively mitigating overfitting issues.

Redundancy Addition: To evaluate the robustness of LLMs against redundancy, we add 0 to 4 redundant premises to each query in the SFT dataset and test the LLMs’ performance on questions with up to 40 redundant premises, with useful information provided in three permutation orders. The evaluation results are presented in Figure 4. Additional results with different model sizes are provided in Appendix A.3.

Figure 4 shows the Vanilla method performs well only on in-distribution samples with Topological order. As redundancy increases in the OOD setting, prediction accuracy declines sharply, indicating overfitting. When SFT with the RC-Aug dataset, the model shows noticeable

performance improvements under in-distribution conditions, likely due to an enhanced capability for reasoning chain generation, as it learns from more reasoning chain data during post-training. Despite this improvement, RC-Aug still struggles to generalize well as the level of redundancy increases. The inference-time scaling method, SCoP- k , results in more consistent performance across surface variations. We observe its significant improvements in OOD settings, including Random and Reversed permutation orders and increased redundancy. However, the improvement is still limited, especially for more challenging arithmetic reasoning task. Additionally, it even shows performance degradation on the Topological dataset, where information is originally provided in order and disturbed in paraphrases.

Results from Figure 4 also shows MEND, consistently outperforms all baselines across different permutation orders and levels of redundancy. These results demonstrate that diversifying question queries for SFT sufficiently enhances the ability of models to capture symmetry information across various surface forms. Consequently, our method not only performs well on in-distribution evaluation

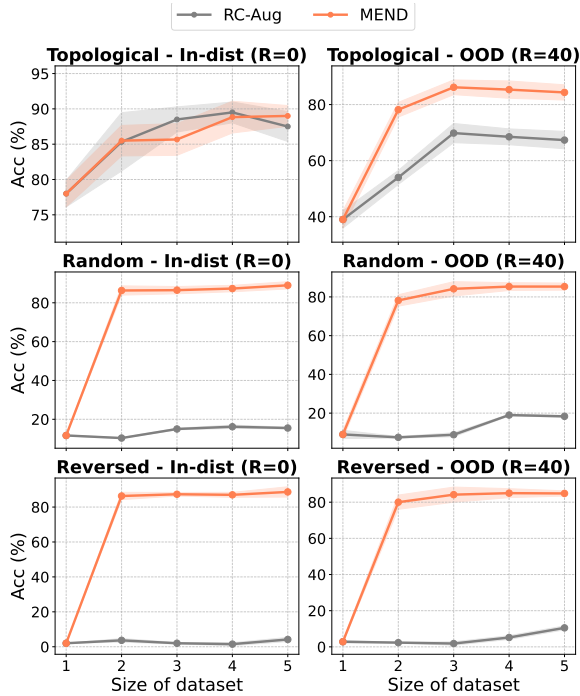


Figure 5: Data efficiency evaluation. R in figure titles indicates the number of redundancy in the query. The size of dataset = 1 indicates using the original dataset for SFT. All plots are averaged among 3 random seeds with temperature= 1. The solid line is the mean value, and the light shade represents the first standard deviation.

samples but also shows strong generalizability to OOD cases with unseen redundancy. We also conduct an ablation study with MEND-RC, where we observe a performance degradation when replacing the data augmentation direction from queries to reasoning chains, highlighting the necessity of enhancing query symmetry information for SFT.

Table 3: VoV evaluation for arithmetic tasks with Llama-3.2-3B model. (\downarrow): The lower, the better for reasoning consistency. We normalize the scale with respect to the variance of Vanilla method. The VoV values for Vanilla are 195.55, and 966.22.

	Vanilla	RC-Aug	SCoP-8	MEND-RC	MEND
VoV_o (\downarrow)	1.00	3.70E+00	3.28E-01	2.03E-01	4.47E-02
VoV_r (\downarrow)	1.00	1.55E+00	3.81E-03	3.53E-03	2.23E-03

Reasoning consistency: We adopt the Variance of Variations (VoV) (Zhou et al., 2024) to quantitatively evaluate the reasoning consistency:

$$\text{VoV}_f = \text{Var}_f(\text{acc}(p)), \mathbf{f} \in \{\mathbf{o}, \mathbf{r}\}, \quad (7)$$

where VoV_o and VoV_r are the variance of prediction accuracy with respect to different permutation

orders and different levels of redundancy. Smaller VoV values indicate better reasoning consistency.

Table 3 shows the VoV values for the arithmetic reasoning task in Figure 4. The Vanilla and RC-Aug method show very large VoV values, indicating poor reasoning consistency and poor capability to capture the symmetry information. The SCoP method successfully reduces the variance since the final results are obtained through majority voting from diverse query paraphrases. Our method also demonstrates strong performance in reducing variance, indicating that we effectively mitigate reasoning consistency issues during post-training without sacrificing inference-time efficiency.

CoT Consistency. To assess the CoT consistency of our method, we use the CoT-consistency score (Wei et al., 2022), defined as the fraction of CoT results that match the majority-vote answer, which is the higher, the better. In our experiments, we fine-tune the models using CoT supervision, and the models also generate CoT reasoning during evaluation before producing the final answer. We report the consistency scores for the Logical Reasoning task using the LLaMA-3.2-1B model in Table 4. We observe that MEND significantly improves reasoning consistency compared to the baselines, showing that MEND can also significantly improve the CoT consistency.

Data Efficiency: We use different amounts of data for SFT and present the results of the arithmetic reasoning task with Llama-3.2-1B model in Figure 5. For the RC-Aug baseline, we observe that increasing the size of the augmented dataset leads to a rapid accuracy improvement in in-distribution conditions but a much slower improvement in OOD settings. This indicates that while RC-Aug can improve performance, it is not efficient in OOD scenarios. In contrast, MEND consistently outperforms the baselines across different amounts of SFT data in OOD settings, demonstrating superior data efficiency and generalizability.

5.4 Reasoning-Consistency Verification

In this subsection, we aim to answer **RQ3** by verifying whether MEND captures the structured information in question queries. We utilize the probing method described in Section 4.4 and report the F1-Macro scores of prediction (4) as the evaluation metric (Hou et al., 2023). A higher score means a better ability to retrieve relevant information from the input queries.

The results are presented in Figure 6. We

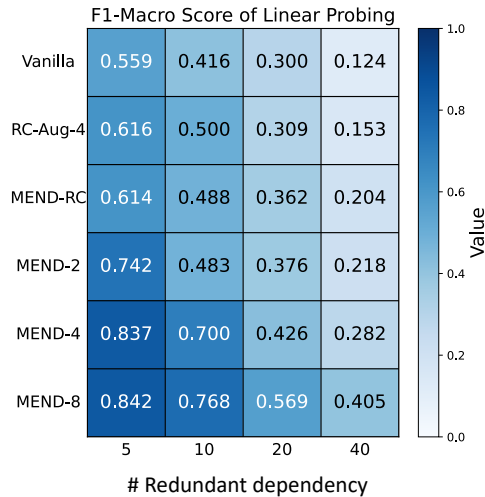


Figure 6: F1-macro score of linear probing on the logical reasoning task with the base model as Llama-3.2-1B. Method- n indicates the dataset size is n times the original one after augmentation.

Table 4: CoT consistency experiment on the Logical reasoning task with base model as Llama-3.2-1B. We compare the performance with different redundancy numbers in the question prompt. **Bold**: the method with the best performance.

# Redundancy		0	10	20	30	40	Avg
Rand	Vanilla	0.972	0.776	0.794	0.721	0.714	0.795
	RC-Aug	0.995	0.963	0.904	0.841	0.816	0.904
	MEND	0.988	0.981	0.979	0.984	0.988	0.984
Reve	Vanilla	0.771	0.758	0.742	0.722	0.724	0.743
	RC-Aug	0.994	0.966	0.915	0.834	0.819	0.906
	MEND	0.986	0.982	0.988	0.989	0.988	0.987
Topo	Vanilla	0.767	0.783	0.734	0.725	0.715	0.745
	RC-Aug	0.995	0.961	0.909	0.857	0.818	0.908
	MEND	0.987	0.983	0.979	0.986	0.991	0.985

observe that the baseline methods RC-Aug and MEND-RC do not show significant improvement in detecting useful information compared to the Vanilla model. In contrast, the proposed method MEND achieves a much higher score than the baselines. This confirms that by enhancing query symmetry, MEND captures more structured information and significantly improves LLMs’ *in-context knowledge extraction* capability for reasoning tasks. The probing results also explain the OOD generalizability improvement of MEND, as it enhances the LLMs’ ability to understand OOD queries. A comparison between our linear probing and KNN probing is provided in Appendix A.4.

6 Conclusion

In this work, we addressed the problem of reasoning equivalence in LLMs with respect to di-

verse surface form variations corresponding to the same semantic meaning. We proposed syMmetry-ENhanceD (MEND) Data Augmentation, enhancing reasoning equivalence by applying structured query-level transformations. Our method improves reasoning robustness at the knowledge extraction stage by enforcing the query symmetry in the SFT dataset. Experiments demonstrate that MEND achieves superior sampling efficiency and generalizability in reasoning tasks. Although one potential risk is that the misuse of this work in real-world scenarios can cause unexpected damage, we hope our findings can provide a foundation for future research on improving LLM reasoning consistency through structured dataset curation.

Limitations

While MEND demonstrates improved performance of reasoning consistency under query variations, we primarily evaluate it on arithmetic and logical reasoning tasks. As a result, its potential and effectiveness on more complex tasks or domains such as specialized scientific reasoning and code generation remain underexplored.

Acknowledgment

The authors would like to thank the collaboration with the Mario Lemieux Center for Heart Rhythm Care at Allegheny General Hospital for supporting this paper.

References

- Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: LLMs trained on "a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*.
- Zhepeng Cen, Yao Liu, Siliang Zeng, Pratik Chaudhari, Huzefa Rangwala, George Karypis, and Rasool Fakoor. 2024. Bridging the training-inference gap in LLMs by leveraging self-generated tokens. *arXiv preprint arXiv:2410.14655*.
- Zhepeng Cen, Yihang Yao, William Han, Zuxin Liu, and Ding Zhao. 2025. [Behavior injection: Preparing language models for reinforcement learning](#). *arXiv preprint arXiv:2505.18917*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

- Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.
- Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.
- Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. 2024. Generating code world models with large language models guided by monte carlo tree search. *arXiv preprint arXiv:2405.15383*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the reversal curse. *arXiv preprint arXiv:2403.13799*.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujie Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.
- Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. *arXiv preprint arXiv:2310.14491*.
- Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.
- Miao Li, Wenhao Ding, Haohong Lin, Yiqi Lyu, Yihang Yao, Yuyou Zhang, and Ding Zhao. 2025. **Crashagent: Crash scenario generation via multi-modal reasoning**. *arXiv preprint arXiv:2505.18341*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujie Yang, Jian Jiao, Nan Duan, et al. 2024a. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*.
- Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujie Yang, Jian Jiao, Nan Duan, Weizhu Chen, et al. 2024b. Not all tokens are what you need for pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew Chi-Chih Yao. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.
- Shiqi Liu, Mengdi Xu, Peide Huang, Xilun Zhang, Yongkang Liu, Kentaro Oguchi, and Ding Zhao. 2023. Continual vision-based reinforcement learning with group symmetries. In *Conference on Robot Learning*, pages 222–240. PMLR.
- Darius Muglich, Christian Schroeder de Witt, Elise van der Pol, Shimon Whiteson, and Jakob Foerster. 2022. Equivariant networks for zero-shot coordination. *Advances in Neural Information Processing Systems*, 35:6410–6423.
- Xinzhe Ni, Yeyun Gong, Zhibin Gou, Yelong Shen, Yujie Yang, Nan Duan, and Weizhu Chen. 2024. Exploring the mystery of influential data for mathematical reasoning. *arXiv preprint arXiv:2404.01067*.
- 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.
- Judea Pearl. 2009. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.
- Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*.
- Stuart Russell and Peter Norvig. 2021. *Artificial Intelligence: A Modern Approach*. Pearson.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. 2025. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*.
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023a. **Large language models can be easily distracted by irrelevant context**. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023b. Large language models

- can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.
- Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. 2024. Drivelm: Driving with graph visual question answering. In *European Conference on Computer Vision*, pages 256–274. Springer.
- Petar Velickovic and et al. 2021. Geometric deep learning: Grids, graphs, groups, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.
- Dian Wang, Robin Walters, and Robert Platt. 2022. So(2)-equivariant reinforcement learning. *arXiv preprint arXiv:2203.04439*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.
- Zhiwei Wang, Yunji Wang, Zhongwang Zhang, Zhangchen Zhou, Hui Jin, Tianyang Hu, Jiacheng Sun, Zhenguo Li, Yaoyu Zhang, and Zhi-Qin John Xu. 2024b. Towards understanding how transformer perform multi-step reasoning with matching operation. *arXiv preprint arXiv:2405.15302*.
- 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.
- Maurice Weiler and Gabriele Cesa. 2019. General e(2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32.
- Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalk, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*.
- Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. 2024a. Equibot: Sim(3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*.
- Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024b. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024a. Physics of Language Models: Part 2.1, Grade-School Math and the Hidden Reasoning Process. *ArXiv e-prints*, abs/2407.20311. Full version available at <http://arxiv.org/abs/2407.20311>.
- Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024b. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv preprint arXiv:2407.20311*.
- Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024c. Physics of Language Models: Part 2.1, Grade-School Math and the Hidden Reasoning Process. *ArXiv e-prints*, abs/2408.16293. Full version available at <http://arxiv.org/abs/2408.16293>.
- 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. *arXiv preprint arXiv:2309.12284*.
- Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhao Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.
- Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh Murthy, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, et al. 2024. Diversity empowers intelligence: Integrating expertise of software engineering agents. *arXiv preprint arXiv:2408.07060*.
- Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*.
- Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. *arXiv preprint arXiv:2404.11500*.
- Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2023. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *The Twelfth International Conference on Learning Representations*.
- Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. 2024a. Dynamic evaluation of large language models by meta probing agents. In *Forty-first International Conference on Machine Learning*.
- Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. 2024b. Dyval 2: Dynamic evaluation of large language models by meta probing agents. *arXiv preprint arXiv:2402.14865*.

A Supplementary Experiments

In this section, we provide some supplementary experiments that are omitted in the main context due to the space limit.

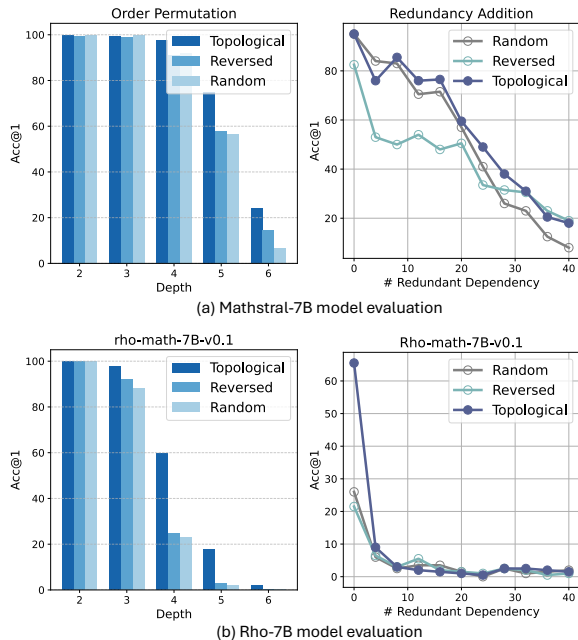


Figure 7: Evaluation of the mathstral-7B and rho-math-7b-v0.1 model.

A.1 Supplementary evaluation of open-source models

In addition to the evaluation of DeepSeek model we presented in section 5.2, we also provide the evaluation results of Mathstral-7B and Rho-math-7b-v0.1. The results of the 8-shot accuracy evaluation are presented in Figure 7. The findings are consistent with those in section 5.2 that LLMs are vulnerable to order permutation and redundancy addition.

A.2 Supplementary results about permutation order experiments

The supplementary experiment results about arithmetic reasoning tasks using Llama-3.2-1B model are shown in Table 5. The observations are basically the same as those in Section 5.3: the proposed MEND achieves the best performance in OOD scenarios against all the baseline methods.

A.3 Supplementary results about redundancy addition experiments

The results for logical reasoning with the base model Llama-3.2-3B and arithmetic reasoning with

Table 5: Accuracy (%) evaluation on datasets with different permutation order and base model as Llama-3.2-1B. The difficulty level is defined by the number of reasoning steps for ground-truth reasoning chains. The number in the parentheses indicates the performance comparison with the vanilla method. **Green**: performance improvement; **Red**: performance degradation. **Bold**: the method with best performance.

Order	Method	Difficulty Level				Avg.
		1	2	3	4	
Topological	Vanilla	99.0	99.0	75.5	23.0	74.1
	RC-Aug	100.0	100.0	84.5	24.5	77.3 (+3.2)
	SCoP-2	100.0	82.5	19.0	2.0	50.9 (-23.2)
	SCoP-4	100.0	92.5	29.0	4.5	56.5 (-17.6)
	SCoP-8	100.0	99.0	40.5	9.0	62.1 (-12.0)
	MEND	100.0	100.0	81.0	11.5	73.1 (-1.0)
Random	Vanilla	99.5	57.0	12.0	1.0	42.4
	RC-Aug	100.0	65.0	12.0	0.5	44.4 (+2.0)
	SCoP-2	100.0	76.5	14.0	2.5	48.3 (+5.9)
	SCoP-4	100.0	94.0	27.5	4.0	56.4 (+14.0)
	SCoP-8	100.0	98.5	38.0	7.5	61.0 (+18.6)
	MEND	100.0	100.0	80.5	7.5	72.0 (+29.6)
Reversed	Vanilla	98.5	8.5	0.0	0.5	26.9
	RC-Aug	100.0	29.0	0.5	0.0	32.4 (+5.5)
	SCoP-2	100.0	79.0	22.5	3.5	51.3 (+24.4)
	SCoP-4	100.0	95.5	30.0	4.5	57.5 (+30.6)
	SCoP-8	100.0	98.5	41.0	3.0	60.6 (+33.7)
	MEND	100.0	100.0	81.0	4.5	71.4 (+44.5)

the base model Llama-3.2-1B are presented in Figure 8. The findings are mostly consistent with those presented in Section 5.3, showing that MEND achieves the best performance. One exception is that the inference-time scaling baseline, SCoP, performs better in the logical reasoning tasks. This may be because the Llama-3.2-3B model is strong enough for this relatively easy task, allowing the inference-time scaling method to achieve good results through majority voting.

A.4 Supplementary experiments about model probing

In Section 4.4, we discuss that the KNN classification method used in (Hou et al., 2023) makes ideal assumptions and faces information aggregation issues. Here, we provide a comparison between KNN classification and the linear probing technique used in this paper. The comparison is presented in Figure 9, where we observe that although KNN probing can reveal differences between tasks with varying levels of redundancy, the F1-Macro score remains relatively low, reducing the confidence of the claims. Additionally, for harder tasks with more redundant information, the performance gap between different methods be-

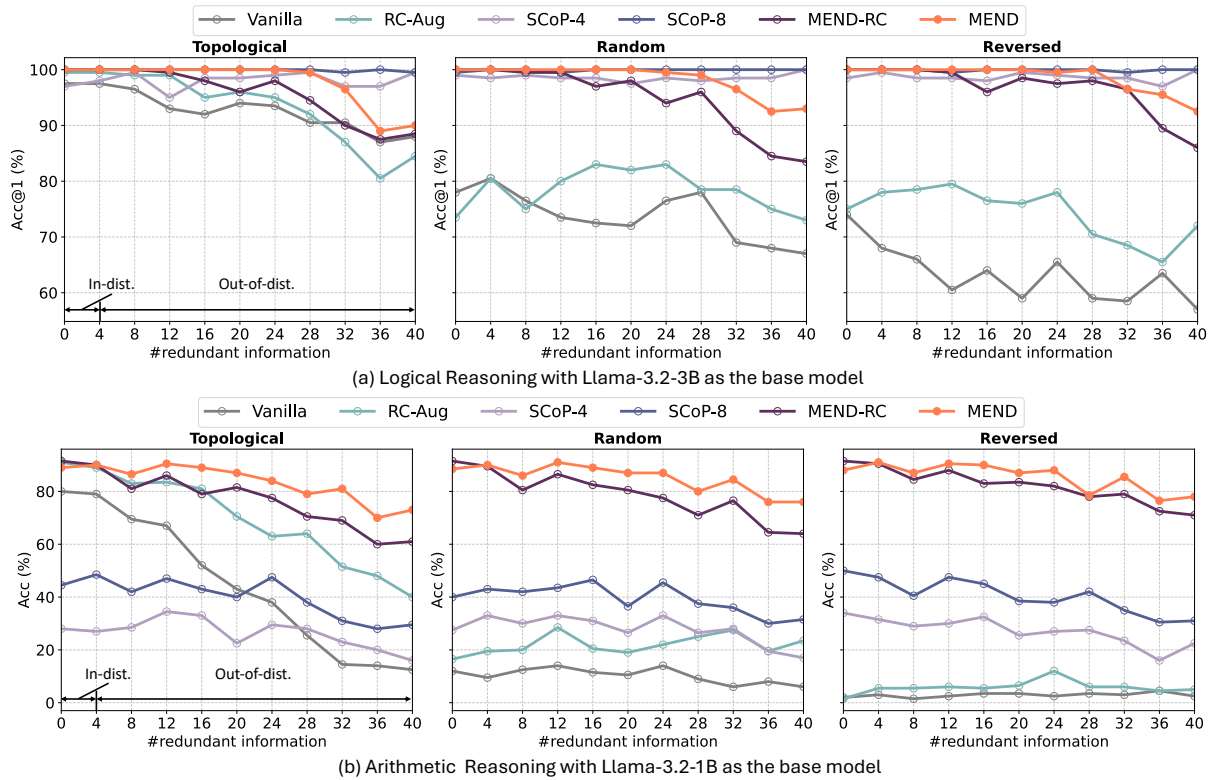


Figure 8: Evaluations with respect to different query variations. Each figure refers to one permutation order type, the x-axis represents the number of redundancies of the test set, and the y-axis represents the accuracy of final answers. For each dataset, we report the accuracy value over a dataset with a size of 200.

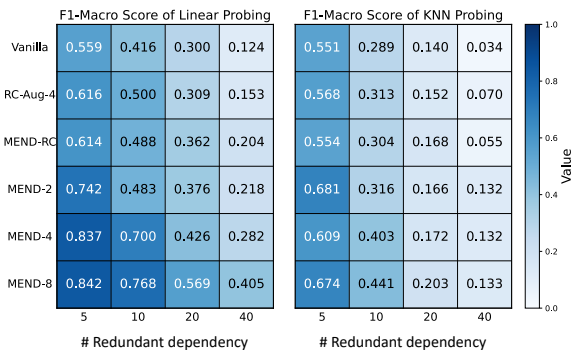


Figure 9: Comparison between linear probing and KNN-based probing used in (Hou et al., 2023). Left: linear probing; Right: KNN-based probing.

comes minor. This is because information tends to aggregate on the beginning tokens, making it difficult to assign appropriate weights to all attention entries for information retrieval probing. Based on this experiment, we reconfirm that our linear probing technique is more suitable for revealing the model’s capability.

A.5 Supplementary evaluation of closed-source models

The Full QA pairs presented in Figure 1 are provided the following code blocks.

Incorrect answer of DeepSeek-V3 to Q1.

```

Let's break down the problem step by step to determine how many pieces of popcorn Gary has to eat.
1. First 30 Seconds:
  - Popped Kernels: 20
2. Next 30 Seconds (30 to 60 seconds):
  - The number of kernels that pop is three times the amount that popped in the first 30 seconds.
  - Calculation:  $(3 \times 20 = 60)$ 
  - Total Popped So Far:
     $(20 + 60 = 80)$ 
3. After Removing from Heat (Residual Heat):
  - A quarter of the kernels that popped in the last 30 seconds of cooking also pop due to residual heat.
  - Calculation:  $(\frac{1}{4} \times 60 = 15)$ 
  - Total Popped So Far:  $(80 + 15 = 95)$ 
4. Next 30 Seconds (60 to 90 seconds):
  - The popping rate increases by 20 from the previous rate.
  - Previous Rate: 60 kernels per 30 seconds
  - New Rate:  $(60 + 20 = 80)$  kernels per 30 seconds
  - Total Popped So Far:  $(95 + 80 = 175)$ 
5. Final 30 Seconds (90 to 120 seconds):
  - The popping rate slows down to half the rate of the past 30 seconds.
  - Previous Rate: 80 kernels per 30 seconds
  - New Rate:  $(\frac{80}{2} = 40)$  kernels per 30 seconds
  - Total Popped So Far:  $(175 + 40 = 215)$ 
Final Calculation:
  - Total Popcorn Popped: 215
However, let's verify the steps to ensure accuracy:
  - First 30 Seconds: 20 kernels
  - Next 30 Seconds:  $(3 \times 20 = 60)$  kernels
  - Residual Heat:  $(\frac{1}{4} \times 60 = 15)$  kernels
  - Next 30 Seconds:  $(60 + 20 = 80)$  kernels
  - Final 30 Seconds:  $(\frac{80}{2} = 40)$  kernels
Adding them up:
 $[20 + 60 + 15 + 80 + 40 = 215]$ 
Therefore, Gary has 215 pieces of popcorn to eat.
 $\boxed{215}$ 

```

Incorrect answer of GPT-4 to Q2.

"In the first 30 seconds, 20 kernels popped. In the next 30 seconds, three times that amount popped, so $20 \times 3 = 60$ kernels popped. In the next 30 seconds, four times the initial popping rate popped, so $20 \times 4 = 80$ kernels popped. In the final 30 seconds, the popping slowed down to half the rate as the past 30 seconds, so $80 / 2 = 40$ kernels popped. After Garrett takes the pan off the heat, a quarter of the number of kernels that popped in the final 30 seconds of cooking also pop from the residual heat, so $40 / 4 = 10$ kernels popped. He believes that five kernels would not have popped if he had not shaken them to the center of the pan, so he has $20 + 60 + 80 + 40 + 10 + 5 = 215$ pieces of popcorn to eat."

B Implementation Details

In this section, we provide additional implementation details about the experiments.

B.1 Simplified Attention

The Simplified Attention (5) was proposed in MechanisticProbe (Hou et al., 2023). To make this paper self-contained, we introduce some necessary details here.

Given a causal language model (LM) with L layers and H attention heads, the attention matrix A is represented as $A = \{A(l, h) \mid 1 \leq l \leq L, 1 \leq h \leq H\}$, where $A(l, h) \in \mathbb{R}^{|T| \times |T|}$. To reduce the size and complexity of A while retaining relevant information for reasoning analysis, we simplify it into A_{simp} using the following steps:

Focus on the Last Token: For causal LMs, attention values directed at the last input token are retained, reducing the size of A to $A_{\text{simp}} \in \mathbb{R}^{L \times H \times |T|}$. This reduction focuses on information most relevant to the final prediction.

Attention Head Pooling: We apply mean pooling across all attention heads to reduce dimensionality further, resulting in $A_{\text{simp}} \in \mathbb{R}^{L \times |T|}$.

Hypernode Simplification (Cross-Statement Attention): For tasks involving multi-token statements, each statement is treated as a hypernode. We apply mean pooling across tokens within each statement and max pooling across question tokens, yielding $A_{\text{cross_simp}} \in \mathbb{R}^{L \times (|S|+1)}$, where $|S|$ represents the number of statements in the query.

These simplification steps ensure that A_{simp} preserves the key information needed for probing reasoning behavior while significantly reducing computational overhead and noise. Full details are in their paper (Hou et al., 2023).

B.2 Experiment Setup Details

The experiments are mainly conducted on the PromptBench benchmark (Zhu et al., 2023). To make the paper self-contained, we explain the data generation process for queries and responses in the experiments. Full details are provided in the documentation of PromptBench.

B.2.1 Query data generation

The promptbench generates the data in two stages: (1) DAG construction; and (2) Natural Language Description of the DAG. An illustration of the DAG is shown in Figure 10.

DAG construction: We generate the DAG with a specified depth and number of redundancy. Based

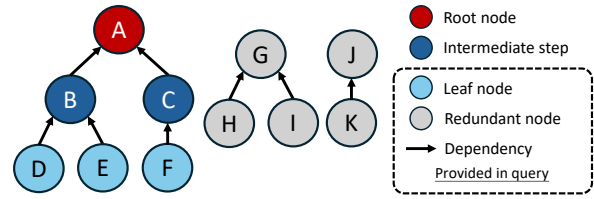


Figure 10: Illustration of the DAG structure: An example DAG with depth=3 and the number of redundant dependency to be 2.

on these, we generate the DAG from top to bottom: Firstly generate the root node, and then sample the dependency between itself and its parent node(s). If the dependency contains a 2-variable operator (e.g. $\{+, -, \wedge (\text{AND})\}$), then it has two parent nodes to generate, otherwise, if it contains a 1-variable operator (e.g. $\{\square^2, \neg (\text{NOT})\}$), it only has one parent node. After the generation of the root node, we then go to the parent generation of its parent node(s), and do this recursively until reach the expected depth. Along with the DAG generation, we also obtain the names of all the generated nodes from a random string generator. After obtaining the DAG, we sample the values for all the leaf nodes from the pre-defined set, and calculate the value of their child nodes from bottom to top.

Natural Language Description After obtaining the DAG and all the node information, we use natural language to describe the question with pre-defined templates. If it is the leaf node, we describe its name and value as:

"The value of node.name is node.value."

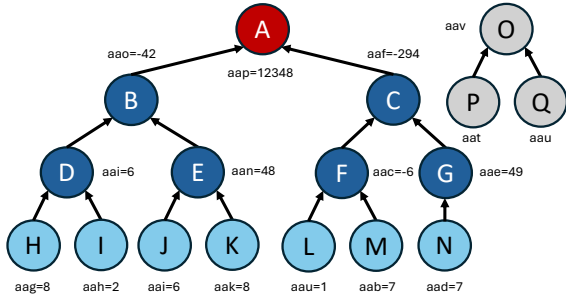
If it is an intermediate node or root node, we describe the dependency with its parent nodes as:

"node.name gets its value by template-func(node.operator, node.parent)."

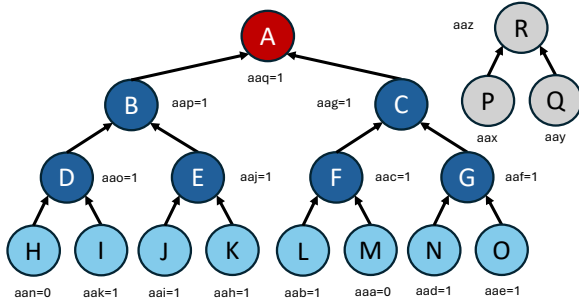
where template-func is a template function to generate strings based on the operator. For example, if the operator is " $-$ ", then the generated string from template-func is:

"subtracting the value of node.parent1.name from the value of node.parent2.name."

If all the information is described in the topological sorting order, then it forms the topological order dataset. If the sentences are shuffled, it forms the random dataset. If the sen-



(a) DAG example in the arithmetic reasoning task



(b) DAG example in the logical reasoning task

Figure 11: DAG Example in the arithmetic reasoning task and the logical reasoning task.

tences are reversed permuted, then it forms the reversed dataset. From the templates above, we can observe that permutating the sentence order in the question prompts does not change the overall semantic meaning of this query.

For the redundancy generation, we describe the redundant nodes and their corresponding dependency based on the process and templates describe above. Since the redundant information is not related to the root node, and their names are not overlapped, describing the redundant information also does not change the semantic meaning of the useful information.

At the end of the problem description, we add one sentence to conclude the question:

"What is the value of root.name?"

We provide example QA pairs for the arithmetic reasoning and logic reasoning tasks and their corresponding DAG structure in this part.

B.2.2 Response generation

In addition to the query mentioned above, SFT data also contains the description of the reasoning chain. As mentioned in section 3, the ground-truth reasoning chain is a topological sorting of this DAG. Similar to the process in query generation, the reasoning chain generation also utilizes some templates. The

difference is that, it contains the answer after every intermediate and final reasoning step. For example:

"node.name gets its value by template-func(node.operator, node.parent), so the value of node.name is node.value."

B.3 Example QA Pairs

For the DAG example shown in Figure 11, the corresponding QA pairs are shown in the following text blocks:

Arithmetic reasoning:

Question:

"The value of aag is 8.
The value of aah is 2.
aai gets its value by subtracting the value of aah from the value of aag.
aav gets its value by multiplying together the value of aat and aaau.
The value of aaj is 6.
The value of aat is 9.
The value of aak is 8.
aan gets its value by multiplying together the value of aaj and aak.
aao gets its value by subtracting the value of aan from the value of aai.
The value of aaau is 1.
The value of aab is 7.
The value of aad is 7.
The value of aaa is 1.
aac gets its value by subtracting the value of aaa from the value of aab.
aae gets its value by squaring the value that aad has.
aaf gets its value by multiplying together the value of aac and aae.
aap gets its value by multiplying together the value of aaf and aao.
What is the value of aap?"

Response:

"aag is 8.0
aah is 2.0
aai = aag - aah = 8.0 - 2.0 = 6.0
aaj is 6.0
aak is 8.0
aan = aaj * aak = 6.0 * 8.0 = 48.0
aao = aai - aan = 6.0 - 48.0 = -42.0
aab is 7.0
aad is 7.0
aaa is 1.0
aac = aab - aaa = 7.0 - 1.0 = -6.0
aae = aad² = (7.0)² = 49.0
aaf = aac * aae = -6.0 * 49.0 = -294.0
aap = aao * aaf = -42.0 * -294.0 = 12348.0
Thus, the answer is 12348.0"

Logical reasoning:

Question:
 "The value 1 means True, and the value 0 means False.
 aak is 1.
 aai is 1.
 aah is 1.
 The value of aaj equals to (aai AND aah).
 aan is 0.
 aax is 1.
 The value of aao equals to (aak OR aan).
 The value of aap equals to (aaj OR aao).
 aab is 1.
 aaa is 0.
 The value of aac equals to (aab OR aaa).
 aad is 1.
 aae is 1.
 The value of aaz equals to (aax AND aay).
 The value of aaf equals to (aad AND aae).
 The value of aag equals to (aaf AND aac).
 The value of aaq equals to (aap AND aag).
 aay is 1.
 What is the value of aaq?"

Response:
 "aak is 1.
 aan is 0.
 $aao = (aak \text{ OR } aan) = (1 \text{ OR } 0) = 1$.
 aah is 1.
 aai is 1.
 $aaj = (aah \text{ AND } aai) = (1 \text{ AND } 1) = 1$.
 $aap = (aao \text{ OR } aaj) = (1 \text{ OR } 1) = 1$.
 aab is 1.
 aaa is 0.
 $aac = (aab \text{ OR } aaa) = (1 \text{ OR } 0) = 1$.
 aad is 1.
 aae is 1.
 $aaf = (aad \text{ AND } aae) = (1 \text{ AND } 1) = 1$.
 $aag = (aaf \text{ AND } aac) = (1 \text{ AND } 1) = 1$.
 $aaq = (aag \text{ AND } aap) = (1 \text{ AND } 1) = 1$.
 Thus, the answer is 1"

B.4 Baseline Implementation Details

- SCoP- k . First, it paraphrases the question prompts k times, then performs reasoning based on these paraphrases, and obtains the final answer through majority voting. Unlike the closed-source LLMs used in (Zhou et al., 2024), our LLMs after SFT lose some of their paraphrasing capability. Additionally, in our experiment, sentence order permutation does not alter the overall semantic meaning, which is a good way for paraphrase. Therefore, we permute the sentences to achieve paraphrasing, resulting in better final performance.
- RC-Aug utilizes the reasoning chain (RC) augmented dataset for SFT, as used in previous work (Yu et al., 2023). Specifically, we augment the answers with different topological orderings while keeping the query unchanged. For a DAG with more than two levels, the

topological ordering is not unique. For example, in the DAG shown in Figure 10, the topological ordering can start by solving either node B first or node C first. When solving node B, it can begin with either node D or node E. Consequently, a fixed DAG can have multiple valid topological orderings, resulting in different reasoning chains that lead to the same correct answer. In RC-Aug, we incorporate these additional reasoning chains from different topological orderings to augment the SFT dataset.

- MEND-RC is an ablation variant that applies MEND to transform the original question queries without introducing new queries into the dataset, while augmenting the dataset with additional reasoning paths.

All the methods include the baselines and MEND utilize full-parameter fine-tuning. We report the accuracy as Pass@1 if not indicated otherwise.

B.5 Computing Resources.

The experiments are run on a server with $2 \times$ AMD EPYC 7542 32-Core Processor CPU, $2 \times$ NVIDIA RTX A6000 graphics, and 252 GB memory. For the arithmetic reasoning tasks with 3B model, it takes about 20 GPU hours for SFT and evaluation. For Logical reasoning tasks with 1B model, it takes about 4 GPU hours for SFT and evaluation.