Order Doesn't Matter, But Reasoning Does: Training LLMs with Order-Centric Augmentation

Anonymous ACL submission

Abstract

Logical reasoning is essential for large language models (LLMs) to ensure accurate and coherent inference. However, LLMs struggle with reasoning order variations and fail to generalize across logically equivalent transformations. LLMs often rely on fixed sequential patterns rather than true logical understanding. To address this issue, we introduce an order-centric data augmentation framework based on commutativity in logical reasoning. We first randomly shuffle independent premises to introduce condition order augmentation. For reasoning steps, we construct a directed acyclic graph (DAG) to model dependencies between steps, which allows us to identify valid reorderings of steps while preserving logical correctness. By leveraging order-centric augmentations, models can develop a more flexible and generalized reasoning process. Finally, we conduct extensive experiments across multiple logical reasoning benchmarks, demonstrating that our method significantly enhances LLMs' reasoning performance and adaptability to diverse logical structures. We release our codes and augmented data in `https://anonymous.4open.science/r/Order-Centric-Data-Augmentation-822C/`.

1 Introduction

Large language models (LLMs) have demonstrated exceptional performance across various real-world applications (Jaech et al., 2024; Dubey et al., 2024; Liu et al., 2024a). Logic reasoning (Cummins et al., 1991) is essential for LLMs. It allows models to draw valid conclusions, maintain coherence, and make reliable decisions across tasks (Pan et al., 2023; Liu et al., 2023).

However, LLMs are sensitive to reasoning order and struggle with logically equivalent transformations (Chen et al., 2024; Berglund et al., 2023b; Tarski, 1956). First, the models are highly sensitive to the order of premises, with perturbing the order

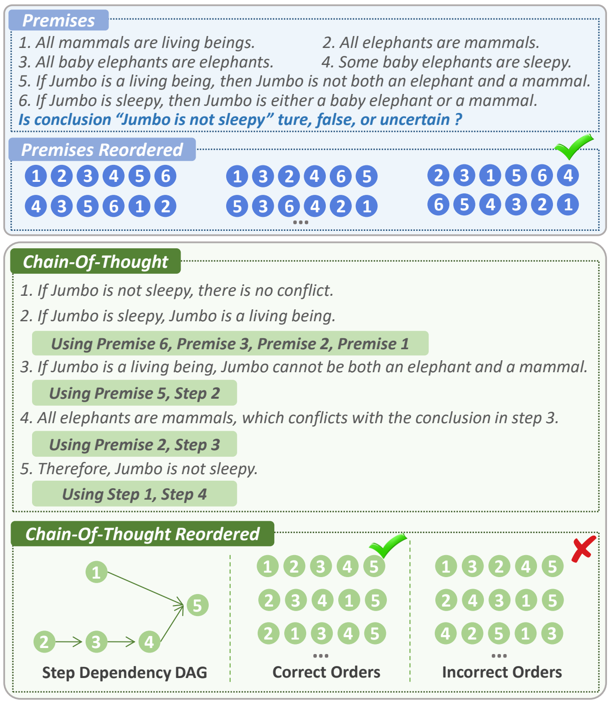

Figure 1: A logical reasoning example. Independent premises can be freely reordered, while reasoning steps must be reordered without violating dependencies.

leading to up to a 40% performance drop (Chen et al., 2024; Liu et al., 2024b). Additionally, if the testing order is reversed compared to the training order, accuracy drops drastically. For example, in the case of data involving two entities within a single factual statement, accuracy drops from 96.7% to 0.1% when training is left-to-right and testing is right-to-left. (Berglund et al., 2023b,a; Allen-Zhu and Li, 2023). This suggests that LLMs follow a rigid logical reasoning order driven by learned patterns rather than true logical understanding.

Existing LLM logical data augmentation methods do not effectively address the sensitivity to equivalent transformations. First, many logical datasets are specifically designed for certain domains, such as specialized fields or exam questions, primarily to broaden the scope of logical reasoning data collection and application (Han et al., 2022; Liu et al., 2020; Yu et al., 2020). Second, a line of work aims to enhance the model's reasoning by mapping natural language to symbolic reasoning (Olausson et al., 2023; Xu et al., 2024; Pan et al., 2023), but it primarily provides symbolic tools for understanding logical language rather than enhancing the logical structure itself. Lastly, another augmentation method creates a "vacuum" world to block interference from real-world logic (Saparov and He, 2022), but it focuses on the impact of the model's prior experience on reasoning, without addressing the design of logical equivalence.

In fact, **commutativity** is a crucial property of logical reasoning. As established by Gödel's completeness theorem (Gödel, 1930) and Tarski's model theory (Tarski, 1956), commutativity means that independent logical units can be freely reordered without changing the essence of the logical structure. Therefore, in logical reasoning, first, independent premises are commutative. As shown in the upper half of Fig. 1, different orders of premises represent equivalent problem structures. Furthermore, as demonstrated by Gentzen's proof theory (Gentzen, 1935), reasoning steps are also commutative, provided their dependencies are intact. As shown in the lower half of Fig. 1, changing the order of steps without disrupting the dependencies results in an equivalent reasoning process. However, altering the order of dependent steps disrupts inference and prevents a coherent path to the correct conclusion.

In this work, we propose an order-centric data augmentation framework that explicitly incorporates logical commutativity into LLM training. For condition order, we randomly shuffle all independent premises. For reasoning steps, we construct a structured, step-by-step reasoning process, identify step dependencies using a directed acyclic graph (DAG), and apply topological sorting to reorder reasoning steps while preserving logical dependencies. Order-centric data augmentation allows models to learn logical equivalence through commutativity, leading to a deeper understanding of logic, rather than relying solely on fixed patterns to solve problems. Our experiments show that order-centric augmentation outperforms training on datasets with a fixed logical structure, enhancing the model's overall reasoning ability and improving its performance in complex shuffled testing scenarios.

Our contributions are summarized as follows: (1) We propose an order-centric logic data augmentation method based on commutativity, which permutes both condition order and reasoning step order, helping models gain a deeper understanding of logical equivalence. (2) We introduce a method that uses DAGs to model the dependencies between reasoning steps, helping to identify valid step reorderings. (3) We conduct extensive experiments to prove the effectiveness of our approach in enhancing logical reasoning.

2 Related Work

2.1 Order Effect of Language Models

Large language models are sensitive to reasoning order. While word order variations in natural language have little impact (Cao et al., 2023; Abdou et al., 2022), disrupting the order in reasoning tasks significantly degrades performance. Chen et al. (2024) show that models perform optimally only when the premise order matches the sequence required for the reasoning process. To address this, Liu et al. (2024b) propose reorganizing premise order to reduce order sensitivity. However, this approach is task-specific and lacks generalizability. Furthermore, the Reversal Curse reveals that models fail to grasp logical equivalence when trained with a fixed linguistic order (Berglund et al., 2023b). Golovneva et al. (2024) mitigate this by proposing reverse training, where LLMs learn both forward and reverse reasoning by randomly shuffling words or segments within a sentence. This highlights the need for diverse training data with varied orderings.

Compared to the above works, we extend to more complex logical reasoning scenarios, building upon this concept by leveraging commutativity for data augmentation in logical reasoning, which helps models generalize across different reasoning structures and enhances robustness.

2.2 Logical Reasoning Enhancing

Existing methods to enhance LLMs' logical reasoning ability mainly fall into three categories: integrating symbolic reasoning, training and inference strategies, and leveraging data augmentation.

Symbolic reasoning enhances LLMs by transforming natural language into formal logic, providing a symbolic approach that helps models understand logic (Olausson et al., 2023; Xu et al., 2024; Zhang et al., 2023). Training and inference

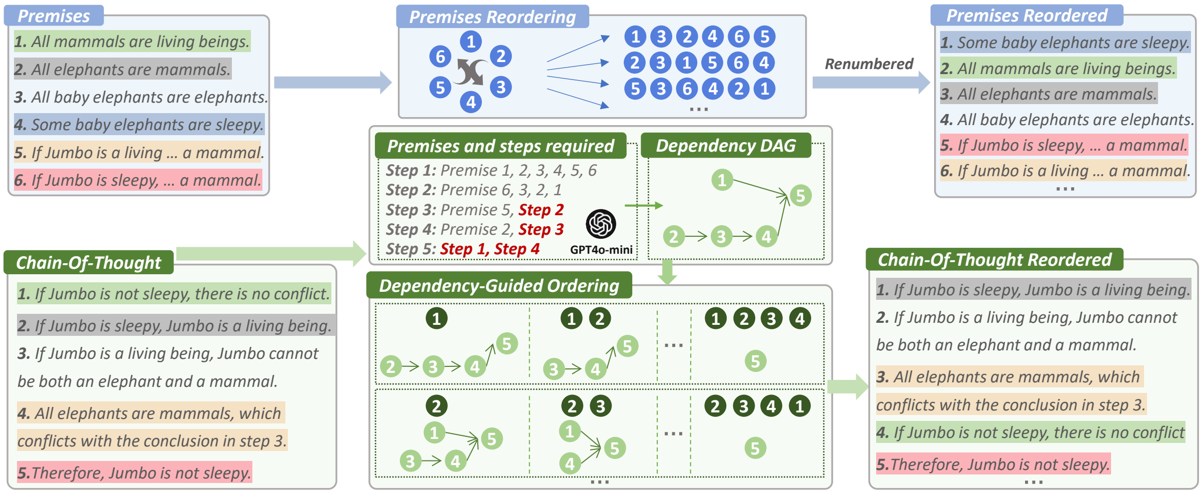

Figure 2: The framework of order-centric data augmentation method. First, we apply condition augmentation by randomly reordering independent premises. Then, we enhance reasoning step order through a directed acyclic graph (DAG) to identify step dependencies and reorder them while preserving logical correctness.

strategies use adversarial pre-training, contrastive learning, and multi-step explicit planning to improve training efficiency and reasoning effectiveness (Pi et al., 2022; Jiao et al., 2022; Zhao et al., 2023). ata augmentation creates diverse training and testing data, aiding models in generalizing better across logical tasks (Han et al., 2022; Tafjord et al., 2021; Clark et al., 2020). LogiGLUE (Luo et al., 2023) builds a large-scale logical benchmark through instruction fine-tuning across deductive, abductive, and inductive tasks. LogicBench (Parmar et al., 2024) focuses on single-rule inference, evaluating LLMs on 25 reasoning patterns and exposing their weaknesses in complex reasoning and negation handling.

Our work falls into the last category. Unlike the previous approaches, we perform order-centric data augmentation on existing logical reasoning datasets, leveraging logical commutativity to enhance the model's understanding of logical equivalence and improve overall logical reasoning ability.

3 Problem Formulation

In this paper, we formulate the problem of logical reasoning in a unified representation. Let $D = \{P, C, L\}$ represent a logical reasoning problem, where $P = \{P_1, P_2, \ldots, P_n\}$ is the set of premises, C is the conclusion, and L is the label, which takes a value from a finite set, such as $\{\text{true}, \text{false}, \text{uncertain}\}$, indicating whether C can be logically inferred from P. In step-based data augmentation, we extend the representation to include a solution $S = \{S_1, S_2, \ldots, S_m\}$, where S consists of reasoning steps that derive the conclusion from the premises. This process can be abstracted as a directed acyclic graph (DAG). Typical logical reasoning datasets only provide labels. Therefore, we construct S ourselves. The specific construction of S will be detailed in Sec. 4.2.

4 Method

In this section, we introduce condition order augmentation in Sec. 4.1 and answer order augmentation in Sec. 4.2. The framework is shown in Fig. 2.

4.1 Condition Order Augmentation

Due to the commutativity of premises, swapping independent premises results in the same solution. Hence, we perturb the order of premises, enabling models to learn the logical equivalence of condition reordering.

4.1.1 Shuffling the Order of Premises

Given a logical reasoning dataset $D_C = \{P, C, L\}$, we first extract the premise set $P = \{P_1, P_2, \ldots, P_n\}$. To generate augmented data, we apply a random permutation σ to the premise set P, producing a new ordered premise set P_{ran}. Specifically:

$$P_{ran} = \{P_{\sigma(1)}, P_{\sigma(2)}, \ldots, P_{\sigma(n)}\}$$

For example, if the original order is $[P_1, P_2, P_3, P_4, \ldots, P_n]$, after applying the permutation σ, the new order might be $[P_3, P_4, P_1, P_n, \ldots, P_2, \ldots]$.

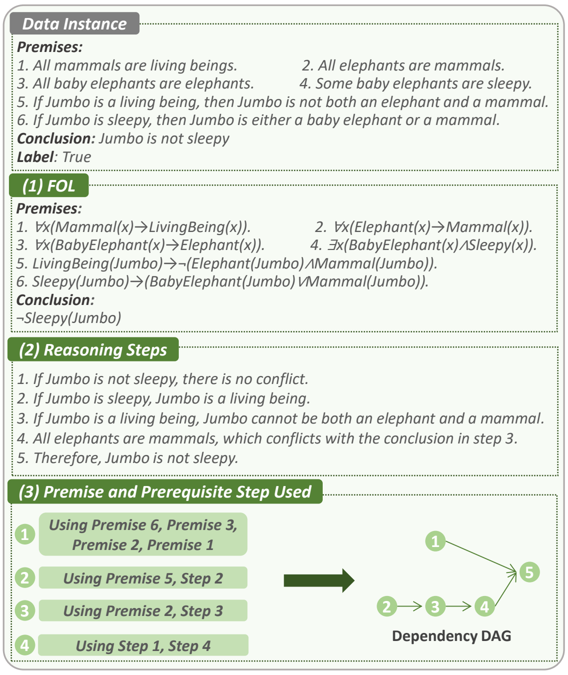

Figure 3: An example of generating a specific solution from data containing only labels and constructing a Directed Acyclic Graph (DAG) to represent the dependencies between steps. Due to space limitations, we only list the conclusions of each step without showing the detailed content.

4.1.2 Generating Augmented Data

We denote the original dataset as $D_C = \{P, C, L\}$ and the augmented dataset as $D'_C = \{P_{ran}, C, L\}$, where P_{ran} represents the randomly shuffled premises. The transformation from D_C to D'_C involves perturbing the order of the premises while keeping the conclusion C and the label L unchanged. Each original data sample generates k instances of condition order augmentation, leading to an augmented dataset D'_C containing $k \times |D_C|$ instances, where $|D_C|$ is the size of the original dataset.

4.2 Answer order Augmentation

Due to the commutativity of reasoning steps, we perturb the order of solution steps to help models learn the logical equivalence of the reasoning process. However, reasoning steps often have dependencies, where the execution of one step may rely on the result of another. To address this, we propose a method for identifying valid step reorderings that ensures these dependencies are preserved.

4.2.1 Leveraging LLMs for Logical Reasoning Solutions

Since logical reasoning datasets typically provide only a single label (e.g., true/false) without a Chain-of-Thought (CoT) reasoning process, we generate detailed step-by-step reasoning solutions to bridge this gap (Xu et al., 2024). We use LLMs[1] for this process. As shown in Fig. 3, the methodology consists of three main steps: (1) For datasets without First-Order Logic (FOL) expressions, We extract their premises and conclusion and convert them into the corresponding FOL representations. (2) The FOL-augmented premises, along with the ground truth labels, are fed into the model, prompting it to generate a step-by-step solution. Each step must clarify its purpose and reasoning, leading to a final conclusion. (3) The generated solutions are then reprocessed by the model to extract the premise indices and prerequisite step indices used in each reasoning step.

4.2.2 Constructing the Step Dependency DAG

After obtaining the logical reasoning solutions, the current data can be represented as $D_S = \{P, C, L, S\}$, where $S = \{S_1, S_2, \ldots, S_m\}$ consists of reasoning steps. We represent S as a directed acyclic graph (DAG), denoted as $G = (V, E)$, where $V = \{S_1, S_2, \ldots, S_m\}$ is the set of reasoning steps, and $E \subseteq V \times V$ is the set of directed edges. An edge (S_i, S_j) indicates that step S_j depends on the result of step S_i.

Each step S_i is represented as a tuple:

$$S_i = (\text{Goal}_i, \mathcal{P}^{(i)}_{used}, \mathcal{S}^{(i)}_{used}, \text{Result}_i)$$

where Goal_i describes the goal of the step, $\mathcal{P}^{(i)}_{used}$ represents the directly used atomic premises, $\mathcal{S}^{(i)}_{used} \subseteq V$ denotes the prerequisite steps that must be executed before S_i, and Result_i is the result derived from the execution of S_i.

4.2.3 Generating Augmented Solution Sequences

A valid reasoning process must maintain all logical dependencies between steps while allowing flexibility in ordering interchangeable steps. We define the dependency constraints as follows:

- A step S_i is **independent** if $\mathcal{S}^{(i)}_{used} = \emptyset$ (i.e., it has no prerequisite steps).

[1] In our experiment, we use GPT-4o-mini.

- A step S_j is **dependent** if $\mathcal{S}_{used}^{(i)} \neq \emptyset$, meaning that it requires prior steps to be completed before execution.

- Two steps S_i and S_j are **order-invariant** if neither step appears in the other's prerequisite set, i.e., $S_i \notin \mathcal{S}_{used}^{(j)}$ and $S_j \notin \mathcal{S}_{used}^{(i)}$.

Our goal is to generate a reasoning sequence that integrates all steps while maintaining dependency constraints, based on the principles outlined above. First, we identify all independent steps where $\mathcal{S}_{used}^{(i)} = \emptyset$ from the dataset, remove them from the DAG, and add them to the list of feasible step sequences *List*. Then, we iterate over all possible combinations of these steps to generate multiple different lists of valid sequences. Next, for each step still present in the DAG, we iterate through the steps of its $\mathcal{S}_{used}$. If $\mathcal{S}_{used}$ contains a step from the current List, we remove that step from $\mathcal{S}_{used}$.

We repeat these two steps until every *List* contains all the steps from S, resulting in a collection of new valid step sequences. We refer to each newly generated sequence as S_{ran}. The final augmented dataset is represented as $D'_S = \{P, C, L, S_{ran}\}$.

Details of the prompts used in order-centric data augmentation are provided in Appendix A.1.

5 Experiments

We conduct experiments to evaluate the effectiveness of our method, focusing on overall performance, training efficiency, and generalization capability.

5.1 Experiment Setup

Datasets (1) **FOLIO** (Han et al., 2022) is a natural language inference dataset annotated with first-order logic (FOL), consisting of 1001 training samples and 231 test samples. (2) **RuleTaker** (Clark et al., 2020) requires models to determine whether a conclusion is entailed by a set of premises, covering various reasoning difficulties. Due to its large scale, we uniformly sample 1000 training and 1000 test instances across different difficulty levels. (3) **LogicNLI** (Tian et al., 2021) is an NLI-style dataset that isolates first-order logic reasoning from commonsense inference for precise logical evaluation. Similarly, we sample 1000 instances from both its training and test sets.

Models We conduct experiments on Llama-3-8B-Instruct (AI@Meta, 2024), Llama-2-13B-Chat (Touvron et al., 2023) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), evaluating model performance under five training conditions: (1) **Untrained**: The original model without any additional training. (2) **Vanilla SFT**: Models fine-tuned only on the original training set, i.e., $D_C = \{P, C, L\}$. (3) **Vanilla SFT + Condition Shuffling**: Models trained on both the original dataset and an augmented version with shuffled condition orders, i.e., $D_C = \{P, C, L\}$ and $D'_C = \{P_{ran}, C, L\}$. (4) **SFT with COT**: Models fine-tuned with training data that includes Chain-of-Thought (COT) reasoning steps, i.e., $D_S = \{P, C, L, S\}$. (5) **SFT with COT + Answer Steps Shuffling**: A model trained with COT data and additional augmentations with shuffled reasoning steps, i.e., $D_S = \{P, C, L, S\}$ and $D'_S = \{P, C, L, S_{ran}\}$.

All models are trained using full fine-tuning, with a 1:1 mix of ShareGPT (Chiang et al., 2023) in each dataset. Training is conducted on four A100 GPUs for four epochs. Each model is trained exclusively on a single dataset, with augmentation applied only to that dataset, and evaluated on the corresponding test set without cross-dataset mixing.

Set	Type	FOLIO	RuleTaker	LogicNLI
Train	Original	1001	1000	1000
	Condition Shuffled	1001	1000	1000
	Answer Step Shuffled	619	594	627
Test	Original	203	2000	2000
	Condition Shuffled	406	2000	2000

Table 1: The data sizes of the training and test sets used in the main experiments.

We applied random shuffling to the premises of each training sample to generate one condition-augmented instance. Due to some data containing multiple valid step orderings, we randomly selected one transformation from each original data to control the data size. Additionally, we shuffled the premises in the test set to create a condition shuffled test set, enabling better evaluation of the model's performance across different logical orders. The data sizes for both the training and test sets are provided in Tab. 1.

5.2 Overall Performance

Tab. 2 shows that our method effectively improves model reasoning performance. Compared

Models	Training	FOLIO		RuleTaker		LogicNLI		Avg.	
		Seq.	Shf.	Seq.	Shf.	Seq.	Shf.	Seq.	Shf.
LLaMA3-8B-Instruct	Untrained	57.64%	55.17%	57.95%	57.95%	28.85%	25.50%	48.15%	46.21%
	Vanilla SFT	63.55%	61.82%	70.65%	68.65%	54.40%	54.90%	62.87%	61.79%
	+ Condition Shuffled	70.44%(+6.89)	67.49%(+5.67)	81.70%(+11.05)	80.15%(+11.50)	59.95%(+5.55)	59.45%(+4.55)	70.70%(+7.83)	69.03%(+7.24)
	SFT with COT	76.35%	73.65%	81.05%	78.80%	42.20%	41.65%	66.53%	64.70%
	+ Answer Steps Shuffled	77.34%(+0.99)	76.85%(+3.20)	84.60%(+3.55)	82.70%(+3.90)	43.80%(+1.60)	42.80%(+1.15)	68.58%(+2.05)	67.45%(+2.75)
LLaMA2-13B-Chat	Untrained	39.16%	35.47%	53.30%	52.50%	28.45%	26.95%	40.30%	38.31%
	Vanilla SFT	53.69%	51.71%	65.35%	62.80%	45.65%	44.20%	54.90%	52.90%
	+ Condition Shuffled	63.05%(+9.36)	62.32%(+10.61)	72.20%(+6.85)	71.30%(+8.50)	54.00%(+8.35)	54.10%(+9.90)	63.09%(+8.19)	62.57%(+9.67)
	SFT with COT	73.40%	70.69%	76.60%	74.65%	43.70%	39.30%	64.57%	61.55%
	+ Answer Steps Shuffled	76.35%(+2.95)	73.89%(+3.20)	75.50%(-1.10)	72.25%(-2.40)	46.90%(+3.20)	42.75%(+3.45)	66.25%(+1.68)	62.97%(+1.42)
Mistral-7B-Instruct-v0.3	Untrained	56.16%	57.39%	54.55%	54.55%	26.55%	24.90%	45.75%	45.61%
	Vanilla SFT	54.68%	55.17%	52.90%	53.20%	25.15%	25.05%	44.24%	44.47%
	+ Condition Shuffled	62.07%(+7.39)	61.82%(+6.65)	70.95%(+18.05)	69.20%(+16.00)	43.00%(+17.85)	44.20%(+19.15)	58.67%(+14.43)	58.41%(+13.94)
	SFT with COT	67.47%	66.75%	81.55%	77.00%	46.40%	44.85%	65.14%	62.87%
	+ Answer Steps Shuffled	72.91%(+5.44)	72.17%(+5.42)	84.10%(+2.55)	82.80%(+5.80)	47.35%(+0.95)	47.30%(+2.45)	68.12%(+2.98)	67.42%(+4.55)

Table 2: The overall performance on FOLIO, RuleTaker, and LogicNLI, where the value in parentheses after each model's Condition Shuffled represents the improvement relative to Vanilla SFT, and the value in parentheses after each model's Answer Steps Shuffled represents the improvement relative to SFT with COT.

to Vanilla SFT, condition shuffling significantly enhances performance. LLaMA3-8B-Instruct achieves +11.05% and +11.5% improvements on RuleTaker's sequential and shuffled evaluations, respectively. Similarly, Mistral-7B-Instruct-v0.3 shows improvements of 17.85% and 19.15% on the two LogicNLI test sets. Overall, the general performance gain ranges from 7% to 15%.

Similarly, adding answer step shuffling further improves performance over COT training. LLaMA3-8B-Instruct achieves +3.55% and +3.90% improvements on RuleTaker's sequential and shuffled evaluations. Mistral-7B-Instruct-v0.3 shows improvements of +5.44% and +5.22% on FOLIO's sequential and shuffled evaluations. Overall, the average performance gain is between 2% and 3%. It is worth noting that LLaMA3-8B-Instruct and LLaMA2-13B-Chat perform worse on the LogicNLI dataset in SFT with COT compared to Vanilla SFT. This could be due to the fact that COT does not always improve model performance across all tasks (Liu et al., 2024c).

Additionally, LLaMA models generally perform 2-3% better on sequential evaluations than shuffled ones, highlighting their sensitivity to order. Nonetheless, condition and answer steps shuffling mitigates this effect. In contrast, Mistral-7B maintains stable performance across both settings, sometimes even outperforming sequential evaluations in shuffled tests.

5.2.1 Training Efficiency

To ensure fairness and exclude the effect of increased data size, we test the accuracy of checkpoints with the same number of training steps, comparing the performance of condition order augmentation with the original order. As shown in Fig. 4, even with the same data size, condition shuffle training consistently outperforms the original order, with the performance gap widening as training progresses. This highlights that the improvement in accuracy is due to the augmentation process itself, rather than the increase in data.

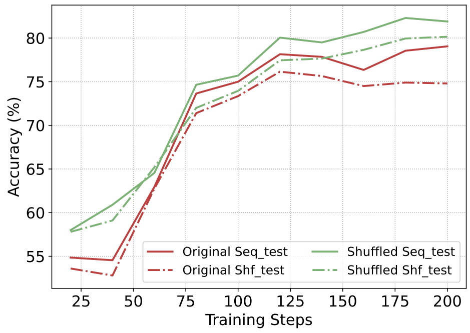

Figure 4: The performance of training efficiency across different training steps in condition order augmentation.

6 Analysis

6.1 Condition Augmentation with Varying Shuffling Degrees

To investigate the effects of premise order transformations, we divide the Kendall tau distance τ between different premise orders and the original order into 10 groups, each spanning a 0.2 range within [-1,1). A τ value of 1 indicates forward order, -1 indicates a complete reversal, and 0 represents a more uniform shuffling. Additionally, random shuffle means that τ values from the entire range may be included. We conduct experiments on RuleTaker using different τ values for condition-based data augmentation.

Model	Test	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	Random
LLaMA3-8B-Instruct	Sequential	69.45%	80.00%	<u>80.55%</u>	75.90%	64.25%	74.80%	69.65%	73.15%	74.40%	67.50%	**81.05%**
	Shuffled	67.55%	77.90%	<u>77.95%</u>	74.85%	64.10%	72.90%	68.50%	70.05%	73.30%	66.20%	**78.80%**
LLaMA2-13B-Chat	Sequential	68.20%	65.25%	71.40%	66.90%	70.60%	60.40%	71.65%	70.35%	72.50%	**73.20%**	<u>72.20%</u>
	Shuffled	67.65%	63.65%	69.60%	65.10%	69.40%	58.75%	69.00%	68.75%	69.55%	<u>70.60%</u>	**71.30%**
Mistral-7B-Instruct-v0.3	Sequential	64.75%	64.80%	54.50%	65.60%	69.05%	50.95%	68.95%	<u>69.65%</u>	67.55%	54.65%	**70.95%**
	Shuffled	64.80%	64.10%	55.55%	66.25%	67.75%	51.15%	<u>67.90%</u>	66.70%	67.40%	54.55%	**69.20%**

Table 3: The performance of the models on RuleTaker with conditionally shuffled premises at different tau values. Tau = 1 represents the original order, tau = -1 indicates a complete reversal, tau = 0 means uniform shuffling, and "Random" refers to a fully random shuffle.

As shown in Tab. 3, random shuffling provides the best performance across all τ values. The level of perturbation in premise order significantly affects model accuracy, with differences exceeding 10%. LLaMA3-8B-Instruct excels with negative τ values, while LLaMA2-13B-Chat and Mistral-7B-Instruct-v0.3 do better with positive τ values. Random shuffled in training data achieves the best overall performance, emphasizing the value of diverse data augmentation for more flexible and robust models.

6.2 The Importance of DAG-based Step Dependency

To explore the importance of using DAG for step dependencies in step augmentation, we use the Answer Step Shuffled data from Tab. 1 as a baseline. We randomly shuffle the steps in the original COT process and assess its performance to evaluate the impact of random step reordering without DAG dependencies.

As shown in Tab. 4, not utilizing DAG dependencies leads to a performance drop compared to DAG-based augmentation. The decline is particularly severe on FOLIO, where LLaMA3-8B-Instruct and LLaMA2-13B-Chat show a drop of 7.64% and 4.68% in the shuffled test. In contrast, Ruletaker and LogicNLI experience a smaller decline.

To explore the underlying cause of this phenomenon, we investigate the degree of dependency between steps in the step dependency DAG. We introduce the **Topological Freedom Index (TFI)**. This metric measures how loosely or tightly connected a DAG is, and it is calculated as follows:

$$TFI = \frac{\text{Number of valid sequences}}{\text{Factorial of the number of steps}} \quad (1)$$

Number of valid sequences represents the number of valid topological orderings that respect the dependency constraints within the DAG. Factorial of the number of steps corresponds the number of

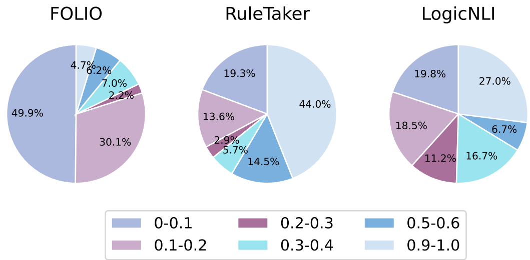

Figure 5: The distribution of TFI index across different intervals in the training sets of FOLIO, RuleTaker, and LogicNLI. Since none of the datasets contain data in the [0.6-0.9) interval, this portion is omitted from the presentation.

possible orderings if no dependencies were present. The closer the TFI value is to 1, the looser the DAG structure, indicating higher reordering flexibility and greater step independence. Conversely, the closer the TFI value is to 0, the stronger the step dependencies, meaning the sequence must follow a strict order with little to no flexibility. Fig. 5 presents the TFI distribution across three datasets, illustrating the degree of step dependency in different reasoning tasks.

In FOLIO, the majority of samples (49.9%) fall within the 0.0–0.1 range, indicating strong dependency constraints and minimal reordering flexibility. In contrast, RuleTaker and LogicNLI exhibit a significantly higher proportion of high-TFI samples (e.g., 44.0% and 27.0% in the 0.9–1.0 range, respectively), suggesting that these datasets contain more loosely connected reasoning structures. These trends highlight that FOLIO imposes stricter logical dependencies. In contrast, the latter datasets have weaker step dependencies, offering greater reordering flexibility. However, this may also imply that the quality of the generated COT needs improvement.

The difference in TFI across datasets aligns with the conclusions we obtained from the Random Step Shuffled experiment. Specifically, stronger step de-

Models	Training data	FOLIO		Ruletaker		LogicNLI	
		Sequential	Shuffled	Sequential	Shuffled	Sequential	Shuffled
LLaMA3-8B-Instruct	Answer Steps Shuffled	77.34%	76.85%	84.60%	82.70%	43.80%	42.80%
	Random Steps Shuffled	76.85%(-0.49)	69.21%(-7.64)	82.10%(-2.50)	81.20%(-1.50)	45.20%(+1.40)	42.75%(-0.05)
	Condition&Answer Shuffled	74.88% (-2.46)	75.86% (-0.99)	81.15% (-3.45)	79.60% (-3.10)	42.80% (-1.00)	43.50% (+0.70)
LLaMA2-13B-Chat	Answer Steps Shuffled	76.35%	73.89%	75.50%	72.25%	46.90%	42.75%
	Random Steps Shuffled	71.92%(-4.43)	69.21%(-4.68)	74.75%(-0.75)	72.75%(+0.50)	43.70%(-3.20)	42.45%(-0.30)
	Condition&Answer Shuffled	70.94% (-5.41)	67.24% (-6.65)	77.40% (+1.90)	73.80% (+1.55)	44.20% (-2.70)	41.60% (-1.15)
Mistral-7B-Instruct-v0.3	Answer Steps Shuffled	72.91%	72.17%	84.10%	82.80%	47.35%	47.30%
	Random Steps Shuffled	71.43%(-1.48)	72.41%(+0.24)	82.95%(-1.15)	79.95%(-2.85)	44.75%(-2.60)	45.25%(-2.05)
	Condition&Answer Shuffled	70.94% (-1.97)	70.44% (-1.73)	82.55% (-1.55)	81.70% (-1.10)	41.50% (-5.85)	42.00% (-5.30)

Table 4: The performance of three different augmentation methods: the first row represents the original DAG-based Answer Steps Shuffled augmentation, the second row represents random step shuffling without dependencies in Sec. 6.2, and the third row represents the combined condition and answer augmentation method in Sec. 6.3.

pendencies result in greater performance loss from random shuffling. Therefore, when performing answer order augmentation, it is important to maintain the integrity of these dependencies.

6.3 Combined Condition and Step Shuffling Leads to Performance Degradation

To investigate the combined effect of condition and step order augmentation, we apply an additional premise shuffle to Answer Steps Shuffled data, adjusting premise references in the answers accordingly. As shown in Tab. 4, **Condition&Answer Shuffled** results in lower performance compared to Answer Steps Shuffled alone.

We believe the key reason is that premise and step shuffling serve different learning purposes. Premise shuffling enables the model to recognize that independent conditions with commutativity can lead to the same answer, while step shuffling allows it to understand that different reasoning paths under the same condition can yield consistent conclusions. When applied separately, each augmentation enhances the model's understanding of logical equivalence, thereby improving its overall reasoning ability. However, when condition and step shuffling are applied together, the logical structure is perturbed in two ways, requiring the model to simultaneously align different orders of both conditions and steps, increasing learning difficulty and reducing generalization. This suggests that excessive augmentation may introduce noise, making it harder for models to establish logical equivalence.

6.4 Effect of Augmentation Frequency

In the main experiment, we set $|D'_C| = |D_C|$, meaning that the parameter $k = 1$. To investigate the impact of augmentation quantity, we increase k and generate $k = 5$ augmented instances for each original training sample in RuleTaker. This leads to an augmented dataset D'_C containing $5 \times |D_C|$ instances. As shown in Tab. 5, adding a few shuffled instances improves model accuracy, but excessive augmentation results in performance degradation. This highlights the need to control the augmentation frequency. The increase in k can lead to a certain degree of performance improvement, indicating that our order-centric data augmentation method has room for further enhancement.

Test	k=1	k=2	k=3	k=4	k=5
Sequential	77.30%	81.35%	**83.45%**	83.15%	77.65%
Shuffled	76.65%	80.35%	81.95%	**83.55%**	79.85%

Table 5: The performance under different augmentation frequencies, where k represents the number of condition order augmentation instances applied per training sample.

7 Conclusion

In this paper, we systematically study how to enhance the logical reasoning ability of LLMs by addressing their limitations in reasoning order variations. We introduce an order-centric data augmentation framework based on the principles of independency and commutativity in logical reasoning. Our method involves shuffling independent premises to introduce order variations and constructing directed acyclic graphs (DAGs) to identify valid step reorderings while preserving logical dependency. Extensive experiments across multiple logical reasoning benchmarks demonstrate that our method significantly improves LLMs' reasoning performance and their adaptability to diverse logical structures.

8 Limitations

Our work primarily focuses on logical commutativity within propositional reasoning tasks. However, this property extends beyond these tasks. It is also prevalent in many other reasoning scenarios, such as mathematical problems and other logic-based tasks. This remains an area for future exploration. Additionally, while we have explored the impact of condition order and answer order augmentations on model performance, how to further integrate and refine these augmentations for better logical reasoning capability is still an open question. We believe our exploration will provide valuable insights for future work on logical equivalence and commutativity in reasoning.

References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919.

AI@Meta. 2024. Llama 3 model card.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*.

Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Max Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. 2023a. Taken out of context: On measuring situational awareness in llms. *arXiv preprint arXiv:2309.00667*.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023b. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*.

Qi Cao, Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Unnatural error correction: Gpt-4 can almost perfectly handle unnatural scrambled text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8898–8913.

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*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *Preprint*, arXiv:2002.05867.

Denise D. Cummins, Todd Lubart, Olaf Alksnis, and Robert Rist. 1991. Conditional reasoning and causation. *Memory and Cognition*, 19(3):274–282.

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

Gerhard Gentzen. 1935. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39(1):176–210.

Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the reversal curse. *arXiv preprint arXiv:2403.13799*.

Kurt Gödel. 1930. Die vollständigkeit der axiome des logischen funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37(1):349–360.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Fangkai Jiao, Yangyang Guo, Xuemeng Song, and Liqiang Nie. 2022. Merit: Meta-path guided contrastive learning for logical reasoning. *arXiv preprint arXiv:2203.00357*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Junjie Liu, Shaotian Yan, Chen Shen, Liang Xie, Wenxiao Wang, and Jieping Ye. 2024b. Concise and organized perception facilitates reasoning in large language models. *Preprint*, arXiv:2310.03309.

Ryan Liu, Jiayi Geng, Addison J Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L Griffiths. 2024c. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. *arXiv preprint arXiv:2410.21333*.

Man Luo, Shrinidhi Kumbhar, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, Chitta Baral, et al. 2023. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models. *arXiv preprint arXiv:2310.00836*.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13679–13707.

Xinyu Pi, Wanjun Zhong, Yan Gao, Nan Duan, and Jian-Guang Lou. 2022. Logigan: Learning logical reasoning via adversarial pre-training. *Preprint*, arXiv:2205.08794.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *Preprint*, arXiv:2012.13048.

Alfred Tarski. 1956. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Oxford University Press.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*.

Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. 2023. Improved logical reasoning of language models via differentiable symbolic programming. *arXiv preprint arXiv:2305.03742*.

Hongyu Zhao, Kangrui Wang, Mo Yu, and Hongyuan Mei. 2023. Explicit planning helps language models in logical reasoning. *Preprint*, arXiv:2303.15714.

A Appendix

A.1 Details of Generating Solutions

In Sec. 4.2, We discuss how to generate step-by-step solutions through $D = \{P, C, L\}$. Specifically, we follow these steps:

(1) For datasets that do not have first-order logic (FOL) expressions, such as RuleTaker and LogicNLI, we extract their premises and conclusions, and use GPT-4o-mini with prompts as shown in Tab. 6 to convert them into corresponding FOL representations. FOLIO, on the other hand, already includes FOL expressions, so no conversion is required.

(2) The FOL-enhanced premises and ground truth labels are input into the model, prompting it to generate step-by-step solutions. As shown in the prompt in Tab. 7, we add two domain-specific examples from each dataset to the prompt, requiring the model to clearly define the purpose and reasoning for each step, eventually leading to the final conclusion. The Task prompt specifies the possible values for the label. Specifically, in FOLIO, the label values are {True, False, Unknown}, in RuleTaker they are {entailment, not entailment}, and in LogicNLI they are {entailment, neutral, self_contradiction, contradiction}.

(3) The model then reprocesses the generated solutions, using prompts like the one shown in Tab. 8, to extract the premise indices and premise step indices used in each reasoning step.

A.2 Kendall Tau Distance

In our study, we investigate the effects of premise order transformations by using the Kendall tau distance τ. This coefficient measures the correlation between two ordered lists, providing a quantitative way to assess how much one order differs from another. We use τ to categorize various permutations of premise orders and assess their impact on model performance.

The Kendall tau coefficient τ is calculated as follows:

$$\tau = \frac{C - D}{\binom{n}{2}}$$

where C is the number of concordant pairs (pairs of items that are in the same relative order in both lists), and D is the number of discordant pairs (pairs that are in opposite order in both lists). The total number of possible pairs is $\binom{n}{2}$, where n is the number of items being compared.

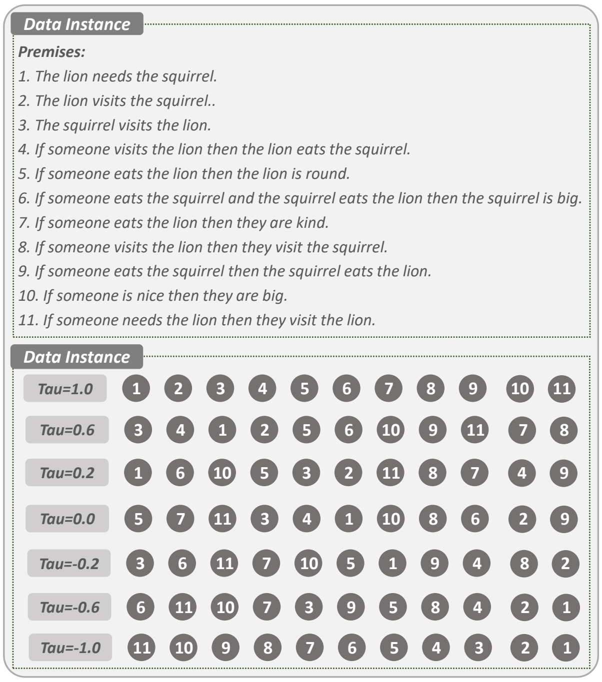

Figure 6: An example of showing the arrangement of premises with different tau values. The tau values do not represent exact values but rather the closest intervals for demonstration purposes.

We divide τ values into 10 groups, each spanning a 0.2 range within the interval [-1, 1). A τ value of 1 indicates that the order of the premises is exactly as required for the reasoning process, while -1 indicates a complete reversal of order. A τ value of 0 indicates that the order is completely random, with no correlation to the original sequence.

For example, if the original premise order is $[P_1, P_2, P_3, \ldots, P_n]$, a permutation function σ might rearrange it to $[P_3, P_1, P_2, \ldots, P_n]$. This process allows us to explore different levels of order perturbation, with the goal of analyzing how such variations affect model performance. Examples of premise orders corresponding to different τ values can be seen in Fig. 6.

/ Task prompt */*
Please parse the context and question into First-Order Logic formulas. Please use symbols as much as possible to express, such as $\forall$, $\wedge$, $\rightarrow$, $\oplus$, $\neg$, etc.

/ Example */*
Premises:
If a cartoon character is yellow, it is from the Simpsons.
If a cartoon character is from Simpsons, then it is loved by children.
Ben is ugly or yellow.
Ramon being real is equivalent to Rhett being not modest and Philip being lazy.
Hypothesis:
James does not have lunch in the company.
Premises-FOL:
$\forall x(Yellow(x) \rightarrow Simpsons(x))$
$\forall x(Simpsons(x) \rightarrow Loved(x))$
$(Yellow(ben) \vee Ugly(ben))$
$real(Ramon) \iff (modest(Rhett) \wedge lazy(Philip))$
Hypothesis-FOL:
$\neg HasLunch(james, company)$

/ Input */*
—INPUT—
Premises:
{**Given_premises**}
Hypothesis:
{**Given_hypothesis**}
—OUTPUT—

Table 6: The prompt for generating First-Order Logic (FOL) expressions corresponding to natural language logical propositions.

/ Task prompt */*
Please solve the question step by step based on First-Order Logic rules such as Modus Ponens, determine whether the hypothesis is true, false, or unknown based on these premises.

/ Example */*
Premises:
1. Walter Folger Brown was an American politician and lawyer who served as the postmaster general.
2. Walter Folger Brown graduated from Harvard University with a Bachelor of Arts.
3. While they were both in Toledo, Walter Folger Brown's father practiced law with Walter Folger Brown.
4. Katherin Hafer married Walter Folger Brown.

Premises-FOL:
1. AmericanPolitician(walterBrown) $\wedge$ Lawyer(walterBrown) $\wedge$ ServedAs(walterBrown, postMasterGeneral)
2. Graduated(walterBrown, harvard) $\wedge$ GraduatedWith(walterBrown, bachelorsOfArt)
3. $\exists t$ (In(walterBrown, toledo, t) $\wedge$ In(walterBrownFather, toledo, t) $\wedge$ PracticedLawTogether(walterBrown, walterBrownFather, t))
4. Married(katherinHafer, walterBrown)

Hypothesis:
Walter Folger Brown was not in Toledo.
Hypothesis-FOL:
$\exists t$ ($\neg$In(walterBrownFather, toledo, t))
Label:
False
Solution:
Step 1: Analyze Walter Folger Brown's presence in Toledo
The third premise states that there exists a time t such that:
In(walterBrown, toledo, t) $\wedge$ In(walterBrownFather, toledo, t) $\wedge$ PracticedLawTogether(walterBrown, walterBrownFather, t).
This means that Walter Folger Brown and his father were both in Toledo at the same time, and they practiced law together there.
Thus, we have clear evidence that Walter Folger Brown was indeed in Toledo at some point.
Step 2: Analyze the hypothesis's claim
The hypothesis states that Walter Folger Brown was not in Toledo, represented in FOL as:
$\exists t$ ($\neg$In(walterBrownFather, toledo, t))
However, this contradicts the third premise, which explicitly states that both Walter Folger Brown and his father were in Toledo at the same time.
Therefore, the hypothesis that Walter Folger Brown was not in Toledo is False based on the premises.
Final Hypothesis:
The hypothesis "Walter Folger Brown was not in Toledo" is False.

/ Input */*
—INPUT—
Premises:
{**Given_premises and premises-FOL**}
Hypothesis:
{**Given_hypothesis and hypothesis-FOL**}
Label:
{**Given_label**}
—OUTPUT—

Table 7: The prompt for generating a step-by-step Chain of Thought (CoT) process based on premises, hypothesis, and label. Different datasets have different sets of labels and examples. For convenience, we only show the prompt on FOLIO here. In practice, we listed two examples, but for brevity and clarity in display, we only present one.

/ Task prompt */*
I will provide you with a description of the question and its answer, and the condition of the question is specific. The answer is done in steps. I hope you can extract the conditions and prerequisite steps used in each step of the answer. Please note that I am not asking you to regenerate the answer yourself, but rather to extract the conditions and prerequisite steps used in each step from the answer I have given you. Meanwhile, the conditions used in the steps are quite explicit, but the prerequisite steps used are quite implicit. I hope you can understand and summarize the prerequisite steps used in each step. Your answer should only include Conditions and prerequisite steps used.

/ Example */*
Question:
Premises:
1. Lana Wilson directed After Tiller, The Departure, and Miss Americana.
2. If a film is directed by a person, the person is a filmmaker.
3. After Tiller is a documentary.
4. The documentary is a type of film.
5. Lana Wilson is from Kirkland.
6. Kirkland is a US city.
7. If a person is from a city in a country, the person is from the country.
8. After Tiller is nominated for the Independent Spirit Award for Best Documentary.
Premises-FOL:
1. DirectedBy(afterTiller, lanaWilson) $\wedge DirectedBy(theDeparture, lanaWilson)$
$\wedge DirectedBy(missAmericana, lanaWilson)$
2. $\forall x \forall y (DirectedBy(x, y) \rightarrow Filmmaker(y))$
3. Documentary(afterTiller)
4. $\forall x (Documentary(x) \rightarrow Film(x))$
5. From(lanaWilson, kirkland)
6. In(kirkland, unitedStates)
7. $\forall x \forall y \forall z ((From(x, y) \wedge In(y, z)) \rightarrow From(x, z))$
8. Nomination(afterTiller, theIndependentSpiritAwardForBestDocumentary)
Conclusion:
Miss Americana is not directed by a filmmaker from Kirkland.
Conclusion-FOL: $\neg \exists x (Filmmaker(x) \wedge From(x, kirkland) \wedge DirectedBy(missAmericana, x))$
Answer:
Step 1: Analyze the premises regarding Lana Wilson's role as a filmmaker
From premise 1, we know that Lana Wilson directed Miss Americana. Therefore, we can conclude that Lana Wilson is a filmmaker based on premise 2, which states that if a film is directed by a person, that person is a filmmaker. Thus, we have: DirectedBy(missAmericana, lanaWilson) $\rightarrow Filmmaker(lanaWilson)$
Step 2: Analyze Lana Wilson's origin
From premise 5, we know that Lana Wilson is from Kirkland. Therefore, we can conclude: From(lanaWilson, kirkland)
Step 3: Combine the information
Since we have established that Lana Wilson is a filmmaker and she is from Kirkland, we can conclude: Filmmaker(lanaWilson) $\wedge From(lanaWilson, kirkland)$
Step 4: Analyze the conclusion's claim
The conclusion states that Miss Americana is not directed by a filmmaker from Kirkland, represented in FOL as: $\neg \exists x (Filmmaker(x) \wedge From(x, kirkland) \wedge DirectedBy(missAmericana, x))$
However, we have already established that Lana Wilson, who directed Miss Americana, is indeed a filmmaker from Kirkland. This directly contradicts the conclusion.
Final Conclusion:
The conclusion "Miss Americana is not directed by a filmmaker from Kirkland" is False.
Conditions and prerequisite steps used:
step 1: condition 1, condition 2
step 2: condition 5
step 3: step 1, step 2
step 4: step 3

/ Input */*
—INPUT—
Question:
{**Given_question**}
Answer:
{**Given_answer**}
—OUTPUT—

Table 8: The prompt for extracting Conditions and prerequisite steps used in each step of step-by-step solutions.