

Premise Order Matters in Reasoning with Large Language Models

Xinyun Chen^{1*}, Ryan A. Chi^{1 2*}, Xuezhi Wang¹ and Denny Zhou¹

*Equal contribution, ¹Google DeepMind, ²Stanford University
{xinyunchen,xuezhiw,dennyzhou}@google.com, ryanchi@cs.stanford.edu

Large language models (LLMs) have accomplished remarkable reasoning performance in various domains. However, in the domain of reasoning tasks, we discover a frailty: LLMs are surprisingly brittle to the *ordering of the premises*, despite the fact that such ordering does not alter the underlying task. In particular, we observe that LLMs achieve the best performance when the premise order aligns with the context required in intermediate reasoning steps. For example, in deductive reasoning tasks, presenting the premises in the same order as the ground truth proof in the prompt (as opposed to random ordering) drastically increases the model's accuracy. We first examine the effect of premise ordering on deductive reasoning on a variety of LLMs, and our evaluation shows that permuting the premise order can cause a performance drop of over 30%. In addition, we release the benchmark R-GSM, based on GSM8K, to examine the ordering effect for mathematical problem-solving, and we again observe a significant drop in accuracy, relative to the original GSM8K benchmark.

✓ **Sorted** → correct

Rules: If duve, then dax. If kere and riff, then tess. If feg and tess, then rabe. If wozz, then riff. If rabe, then thoxe. If ag and riff and kere, then hepe. If thoxe, then vepe. If nife and hume and ag, then dax. If feg and thoxe, then ire. If rebe and ene and heff, then kone. If ire and dax, then wope. If tess and vepe, then nup. If rabe and vide and nife, then ag. If nup and ag and vide, then duve. If zeck and hepe and dax, then riff. If nup, then hepe. If sene, then hume. If hepe and tess and ag, then kere. If fum, then vide. If kere and wope, then fum. If jag and kone, then thoxe. If fum, then wozz.

Facts: Alice is cote. Alice is kone. Alice is duve. Alice is ag. Alice is jag. Alice is tess. Alice is riff. Alice is feg. Alice is vide.

Query: Is Alice wozz? In each step, indicate whether you are using a fact, or a rule.



... since Alice is fum, by the rule "If fum, then wozz," Alice is wozz. Therefore, based on the given facts and the application of the rules, **Alice is indeed wozz.**

✗ **Shuffled** → incorrect

Rules: If nup, then hepe. If kere and riff, then tess. If feg and tess, then rabe. If wozz, then riff. If tess and vepe, then nup. If ag and riff and kere, then hepe. If feg and thoxe, then ire. If nife and hume and ag, then dax. If ire and dax, then wope. If rebe and ene and heff, then kone. If hepe and tess and ag, then kere. If rabe, then thoxe. If rabe and vide and nife, then ag. If fum, then wozz. If zeck and hepe and dax, then riff. If kere and wope, then fum. If sene, then hume. If thoxe, then vepe. If fum, then vide. If duve, then dax. If jag and kone, then thoxe. If nup and ag and vide, then duve.

Facts: Alice is cote. Alice is kone. Alice is duve. Alice is ag. Alice is jag. Alice is tess. Alice is riff. Alice is feg. Alice is vide.

Query: Is Alice wozz? In each step, indicate whether you are using a fact, or a rule.



... we haven't established a clear path to "fum" based on the given facts and rules... based on the provided facts and the logical steps followed, **we cannot conclude that Alice is wozz.**

Figure 1 | Premise order affects the reasoning performance: a failure case for logical reasoning. Left: rules are sorted in the same order as the ground truth proof (forward order with $\tau = 1$ as defined in Section 2.1). Right: the wrong prediction with GPT-4-turbo after shuffling the rule set ($\tau = 0$). Distracting rules are in bold and light blue.

1. Introduction

Large language models (LLMs) have demonstrated impressive performance across a variety of reasoning tasks (Austin et al., 2021; Chen et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021; Wei et al., 2022). In particular, recent state-of-the-art LLMs have reached or even surpassed human performance on multiple reasoning benchmarks, including STEM problem-solving and code generation (Bubeck et al., 2023; Gemini, 2023; Li et al., 2022). However, recent works show that LLMs exhibit failure modes that align with human-like cognitive bias (Berglund et al., 2023; Hagendorff et al., 2023; Jones and Steinhardt, 2022; McCoy et al., 2023; Shi et al., 2023). For example, Berglund et al. (2023) revealed the *Reversal Curse*; i.e., LLMs trained on “A is B” tend to fail to infer that “B is A.” Distractibility is another failure mode (Jones and Steinhardt, 2022; Shi et al., 2023), where the LLM performance drastically decreases when irrelevant context is included in the task description.

In this work, we investigate the effect that premise order has on LLM reasoning. Specifically, in deductive reasoning, changing the order of premises alone does not change the conclusion. Consider the following illustrative example:

1. If A then B.
2. If B then C.
3. A is True.

We can derive that C is True regardless of the order of these 3 premises. While some studies show that humans have a preference on the premise order to facilitate their reasoning (Dekeyser et al., 2000; Girotto et al., 1997), the premise order does not drastically affect human performance, especially for problems that only involve *modus ponens* (if P then Q; P; therefore Q), which are relatively straightforward for humans.

In contrast to humans, we observe that for LLMs, the premise order has a significant impact on reasoning performance. In particular, LLMs reach the best performance when the premises are arranged **in the same order** as they appear in the ground-truth proof. Taking the illustrative problem above as an example, we observe two phenomena:

1. Presenting “If A then B” before “If B then C” in the prompt generally achieves a higher accuracy compared to the reversed order.
2. The performance gap is more significant when the number of premises increases.

Intuitively, such a preference on the premise order aligns with human preference (Dekeyser et al., 2000) because in the preferred order, each derivation step can be done on-the-fly while looking at premises one by one, without needing to look back and forth across all premises at each step.

We conduct a systematic study on the premise order effect using a variety of SoTA LLMs, including GPT-4-turbo, GPT-3.5-turbo (OpenAI, 2023), PaLM 2-L (Google, 2023), and Gemini 1.0 Pro (Gemini, 2023). Our primary focus is deductive reasoning, and we benchmark all LLMs on problems that only involve *modus ponens* (if P then Q; P; therefore Q), where all LLMs in our evaluation at least achieve decent performance with a small number of premises. We show that the accuracy decrease caused by different ordering can be more than 30%. The ordering effect is further amplified when irrelevant premises (i.e., premises that are not needed to derive a conclusion) are presented in the prompt. Figure 1 illustrates a failure case, where all LLMs fail to generate the proof after changing the order of relevant rules. Interestingly, while all LLMs perform best when the premise order follows the ground truth proof, they reveal different preferences on other alternative orderings. Specifically, compared to randomly ordering the premises, GPT-4-turbo and GPT-3.5-turbo generally achieve better

performance when the premise order is exactly the reverse of the ground truth proof, which enables LLMs to perform derivation via backward chaining. On the other hand, PaLM 2-L generally achieves the **worst performance** with such a reversed order.

Besides logical reasoning, we construct R-GSM to further investigate the ordering effect on mathematical reasoning. Specifically, we build R-GSM on top of a subset of GSM8K experiments, where we change the order of sentences in the problem description and manually verify that the ground truth answer remains the same. Our experiments again show that the performance of all LLMs notably drop, especially on longer problems that require more reasoning steps.

Our evaluation highlights that even in reasoning domains where the premise order **does not matter**, premise order **does matter in LLM reasoning**. Specifically, the premise ordering effect indicates that LLMs are more comfortable reasoning via reading left-to-right instead of back-and-forth, which can be attributed to the auto-regressive model design or the reasoning bias learned from the training corpus. We leave proposing new training and modeling techniques to mitigate the premise order effect as future work.

2. Benchmarks

2.1. Logical Reasoning

Prior work has revealed the weaknesses of LLMs in logical reasoning (Han et al., 2022; Saparov and He, 2022; Saparov et al., 2023; Wan et al., 2024; Xu et al., 2023; Yan et al., 2023), especially when the proof is long and requires the knowledge of multiple deduction theorems. To isolate the effect of premise orders, we focus on a confined problem space adapted from SimpleLogic (Zhang et al., 2022), which only includes propositional logic problems with definite clauses. Specifically, each problem includes: (1) a set of facts A_1, \dots, A_n that hold true; (2) a set of rules of the form “If X , then Y ”, “If X_0 and X_1 , then Y ”, or “If X_0 and X_1 and X_2 , then Y ”; and (3) a conclusion “ C is True” to be proved. As opposed to SimpleLogic — which formulates the problem as a binary classification task (i.e., indicate whether the conclusion is True or False) — in our benchmark, every problem has a ground-truth label of True, and we consider the prediction to be correct only when the generated proof is completely valid. With these strict criteria, the LLM is required to produce the step-by-step deduction that leads to the conclusion, and any hallucination of non-existent facts and rules is considered erroneous.


The key characteristic of our benchmark is that for each logical reasoning problem, we **synthetically generate variants with different premise orders**. Specifically, we denote the order that conforms to the ground truth proof with forward chaining as the *forward* order, where the rule applied in each derivation step is sequentially presented in the problem description. Intuitively, presenting premises in the forward order simplifies the problem for humans, as this allows us to write the proof on-the-fly while reading the premises. Conversely, a premise ordering that is more random increases the task difficulty, since carrying out the derivation requires us to repetitively look for premises for each reasoning step. Motivated by this intuition, we categorize different premise orders based on their Kendall tau distance τ (Cicirello, 2019; Sen, 1968) to the forward order, normalized into the range $[-1, 1]$. Specifically, $\tau = 1$ is the *forward* order, and we denote the order with $\tau = -1$ as the *backward* order, which is the reverse of the forward order and aligns with the proof via backward chaining. $\tau \approx 0$ suggests that there is no strong correlation between the premise order in the problem description and the proof. To thoroughly investigate the LLM preference on different premise orders, we evaluate the model performance on $\tau = 0.5$, 0 and -0.5 , in addition to the forward ($\tau = 1$) and backward ($\tau = -1$) orders. We present examples with $\tau = 1$ and 0 in Figure 1, and defer examples with other τ values to Figure 11 in Appendix B.

We measure the premise order effect by varying the following two factors:

- **Number of rules required in the proof.** It is expected that the premise order effect is more significant with more rules. For our benchmark, we generate problems whose numbers of rules range from 4 to 12.
- **Number of distracting rules** (i.e., rules that are not useful for the proof) presented in the problem. The presence of distracting rules also complicates the problem, as premise selection itself is challenging (Ferreira and Freitas, 2020; Irving et al., 2016; Wang et al., 2017), and LLMs are shown to be easily distracted by irrelevant context (Shi et al., 2023). We include problem variants with 0, 5 and 10 distracting rules.

We generate 200 problems for each number of required rules. Considering different premise orders and numbers of distracting rules, each problem includes 15 variants, resulting in a total of 27K problems in our benchmark.

2.2. R-GSM for Mathematical Reasoning


 **Original → correct**

Thomas withdraws \$1000 in 20 dollar bills from the bank account. **He loses 10 bills while getting home.** After that, he uses half of the remaining bills to pay for a bill. Thomas then triples his money. He then converts all his bills to 5 dollar bills. How many 5 dollar bills does he have?

Let's break down the problem step by step:

1. Thomas withdraws \$1000 in \$20 bills.
 $\$1000 / \$20 = 50$ bills
- 2. He loses 10 of those bills. $50 \text{ bills} - 10 \text{ bills} = 40 \text{ bills}$**
3. He uses half of the remaining bills to pay for a bill. Half of 40 bills is 20 bills. $40 \text{ bills} - 20 \text{ bills} = 20 \text{ bills}$ remaining
4. Thomas then triples his money. He has 20 bills of \$20, which is $20 * \$20 = \400 . Tripling \$400 gives us $\$400 * 3 = \1200 .
5. He converts all his money to \$5 bills.
 $\$1200 / \$5 = 240$ bills

So, Thomas has 240 five-dollar bills.

 **Reordered → incorrect**

Thomas withdraws \$1000 in 20 dollar bills from the bank account. After getting home, he uses half of the remaining bills to pay for a bill. Thomas then triples his money. He then converts all his bills to 5 dollar bills. **He lost 10 bills while getting home.** How many 5 dollar bills does he have?

Let's break down the problem step by step:

1. Thomas withdraws \$1000 in \$20 bills from the bank. $\$1000 / \$20 = 50$ bills
2. He uses half of the remaining bills to pay for a bill. Half of 50 bills is 25 bills. So, he uses 25 bills and has 25 bills left.
3. Thomas then triples his money. Since he had 25 bills of \$20 each, he had $25 * \$20 = \500 before tripling his money. Tripling \$500 gives him $\$500 * 3 = \1500 .
4. He then converts all his bills to \$5 bills.
 $\$1500 / \$5 = 300$ bills
- 5. He lost 10 bills while getting home. $300 \text{ bills} - 10 \text{ bills} = 290 \text{ bills}$**

So, Thomas has 290 \$5 bills.

Figure 2 | R-GSM example where the original problem can be correctly solved by all LLMs in our evaluation, but all of them failed on the reordered one. Different calculation steps and their corresponding problem statements are annotated in light blue. Specifically, the reasoning steps of the original problem follows the ordering of problem statements, while the reordered problem does not.

To further assess the effect of premise orders beyond logical reasoning, we construct the R-GSM dataset based on GSM8K (Cobbe et al., 2021), which is a popular benchmark of grade school math word problems. Specifically, we first select GSM8K test problems with at least 5 sentences in the problem description, then filter out those problems where there is no alternative ordering that does not change the ground truth answer, e.g., problem statements that follow the causal order of an event series. For each of the remaining problem, we keep the last sentence untouched and rewrite the problem description with a different ordering of other sentences. Minor editing on words is allowed to ensure the grammatical correctness of the problem description. To facilitate the annotation

process, for each problem, we write a simple function to enumerate all alternative orderings of problem statements until an ordering that causes the LLM prediction failure is discovered, which can be used for our manual rewriting if the alternative ordering found in the enumeration process happens to preserve the ground truth answer. In total, our R-GSM benchmark contains 220 pairs of problems, including both the original GSM8K problem description and the manually rewritten one with a different ordering of problem statements. Despite that over 60% of problems in R-GSM only have 5 sentences, and all problems have at most 8 sentences, our evaluation shows that all LLMs still perform considerably worse on rewritten problems. Figure 2 presents an example in R-GSM where all LLMs correctly solve the original problem but not the rewritten one. Specifically, the reasoning steps for the original problem follows the ordering of problem statements, while for the rewritten problem, the second calculation step in the correct solution should refer to the second-to-last sentence instead of the second sentence in the problem description. We provide a more detailed case study in Section 3.3, and present the full dataset statistics in Appendix A.

3. Experiments

3.1. Experimental Setup

We evaluate the premise ordering effect on GPT-4-turbo, GPT-3.5-turbo, PaLM 2-L and Gemini 1.0 Pro. We perform the greedy decoding with the temperature 0, and apply the zero-shot prompting in all experiments. On R-GSM, the model input only contains the problem description without additional instructions. For logical reasoning, as shown in Figure 1, we add an instruction in the prompt to ask for a derivation that specifies which premise is used in each step.

3.2. Logical Reasoning

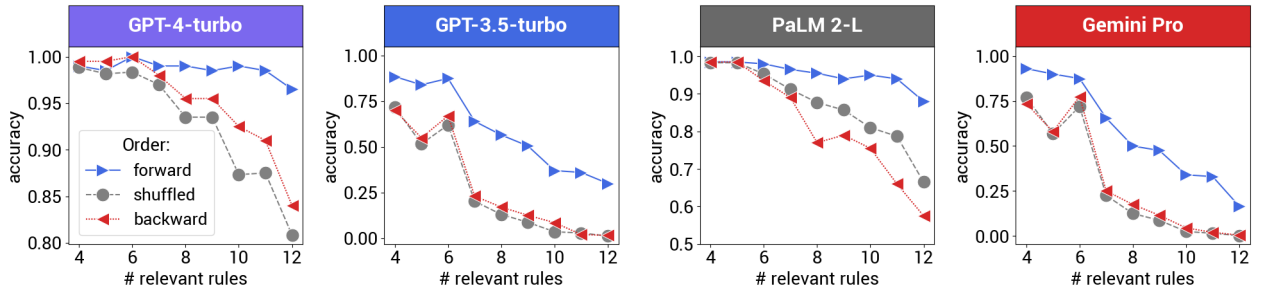


Figure 3 | Logical reasoning without distracting rules. See Table 6 in Appendix E for accuracy numbers.

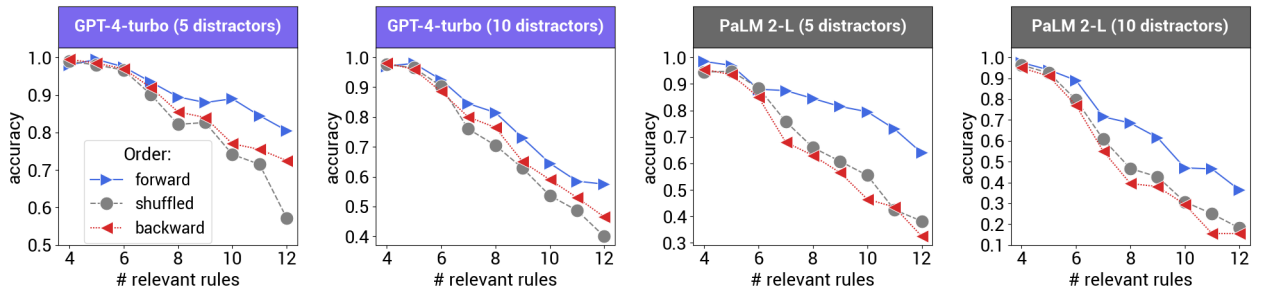


Figure 4 | Logical reasoning with distracting rules. See Tables 7 and 8 for accuracy numbers.

Figure 3 presents the results with different numbers of relevant rules included in ground truth proofs, where the problem does not contain distracting rules, and the shuffled accuracy is the aggregation of

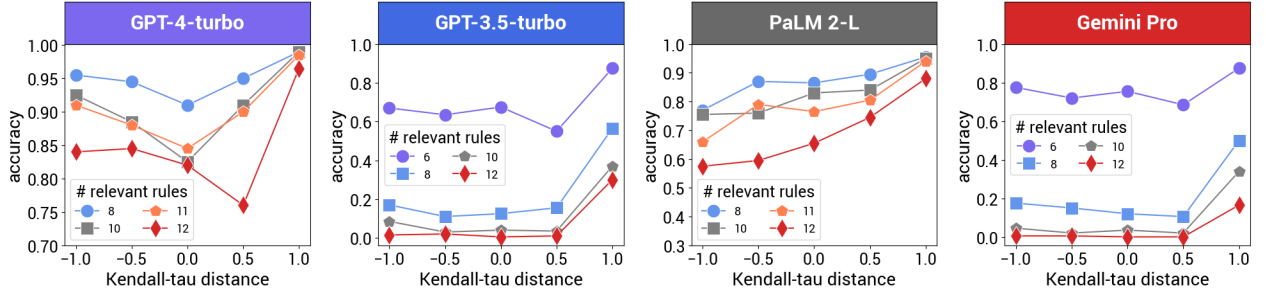


Figure 5 | Results on different τ without distracting rules. See Table 9 for accuracy numbers.

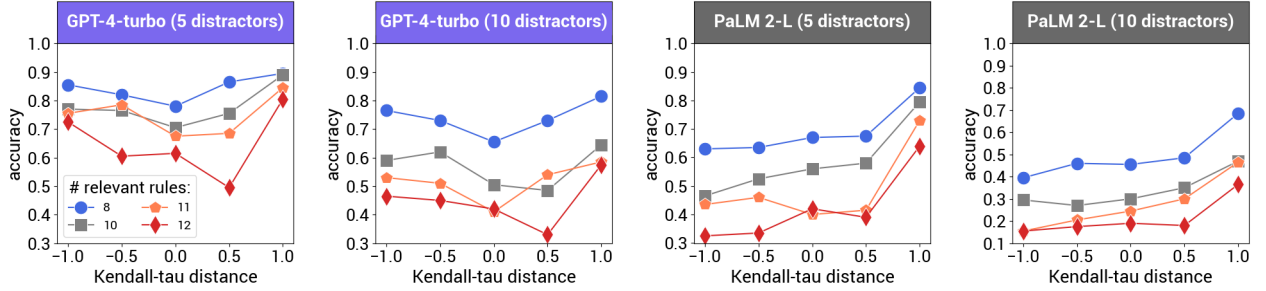


Figure 6 | Results on different τ with distracting rules. See Tables 10 and 11 for accuracy numbers.

results with $\tau = 0.5, 0$ and -0.5 . Across different LLMs, the forward order consistently achieves the best performance, which aligns with the human preference. The performance drop caused by alternative orderings becomes more significant when the number of rules increases. Meanwhile, models with weaker reasoning capabilities are also more sensitive to different premise orders. Specifically, while the accuracy decrease of GPT-4-turbo and PaLM 2-L is up to 20 – 30%, with Gemini 1.0 Pro and GPT-3.5-turbo, changing the premise order from the forward order can degrade the accuracy from over 65% to below 25%, with an accuracy decrease of more than 40%.

Breakdown on different premise orders. We present the results of fine-grained breakdown on premise ordering in Figure 5, where the orders are categorized based on Kendall tau distance τ as described in Section 2.1. Interestingly, while the top preference of all LLMs is the forward order, their preferences on other orders are not alike. Specifically, GPT-4-turbo generally prefers the backward order over other orders, and the overall performance decreases with a smaller absolute value of τ . This observation is also consistent with the human reasoning pattern, as backward chaining is another well-established inference method. On the other hand, PaLM 2-L generally performs the worst with the backward order. With the decrease of τ (i.e., the premise order deviates more from the forward order), the accuracy drops. The preferences of Gemini 1.0 Pro and GPT-3.5-turbo are less consistent, still they prefer the backward order more often than other non-forward premise orders.

Effect of distracting rules. We assess the effect of distracting rules of GPT-4-turbo and PaLM 2-L, which reach a decent performance without the presence of distracting rules. Figures 4 and 6 show that adding distracting rules further decreases the reasoning performance and magnifies the effect of different premise orders. Still, the overall preferences of both LLMs remain the same as the scenario without distracting rules. Specifically, both LLMs again achieve the best performance with the forward order, and GPT-4-turbo prefers the backward order over other non-forward orders, while PaLM 2-L performance decreases with a smaller τ .

Error analysis. In Table 1, we present the breakdown on prediction errors with different premise orders. We consider the following error categories:

1. *wrong refutation*: the LLM wrongly claims that the conclusion can not be proved;
2. *rule hallucination*: the LLM generates rules that do not exist in the problem;
3. *fact hallucination*: the LLM generates facts that do not exist in the problem and are unproven.

We observe that for all LLMs, fact hallucination is typically the most common error pattern, and this error type escalates dramatically with the decrease of τ . The main reason is that LLMs are inclined to use the rules in the sequential order as they present in the problem, so when the next rule in the problem is not yet applicable, LLMs might still hallucinate facts to complete the proof step. Simultaneously, we observe that the percentage of wrong refutation is generally lower for $\tau = -1$ than for $|\tau| < 1$. We present an example of wrong refutation in Figure 1, and we include more examples of rule and fact hallucination in Figure 10 of Appendix B.

	τ	Correct	Wrong Refutation	Hallucination Rule	Hallucination Fact
GPT-4-turbo	1	96.5%	0.5%	1.5%	1.5%
	0.5	76.0%	10.5%	2.0%	11.5%
	0	82.0%	4.5%	3.5%	10.0%
	-0.5	84.5%	1.0%	4.5%	10.0%
	-1	84.0%	0.0%	3.5%	12.5%
GPT-3.5-turbo	1	30.0%	24.5%	9.5%	35.5%
	0.5	1.0%	54.5%	9.5%	33.0%
	0	0.5%	55.0%	7.5%	34.5%
	-0.5	2.0%	50.0%	8.5%	37.5%
	-1	1.5%	34.5%	14.5%	47.0%
PaLM 2-L	1	88.0%	0.5%	3.0%	8.5%
	0.5	74.5%	1.5%	9.5%	14.5%
	0	65.5%	2.0%	11.0%	21.5%
	-0.5	59.5%	1.5%	10.0%	29.0%
	-1	57.5%	1.0%	11.5%	30.0%
Gemini 1.0 Pro	1	16.5%	28.0%	5.0%	50.5%
	0.5	0.0%	59.0%	3.5%	37.5%
	0	0.0%	34.0%	9.0%	57.0%
	-0.5	0.5%	24.5%	9.5%	65.5%
	-1	0.5%	27.5%	11.5%	60.5%

Table 1 | Error analysis for logical reasoning with 12 relevant rules and no distracting rules.

3.3. R-GSM for Mathematical Reasoning

	Init Acc	Reorder Acc		Init Acc	Reorder Acc
GPT-4-turbo	94.1%	85.0%	GPT-4-turbo	100%	89.9%
PaLM 2-L	86.4%	79.5%	PaLM 2-L	100%	87.9%
Gemini 1.0 Pro	80.5%	69.1%	Gemini 1.0 Pro	100%	74.6%
GPT-3.5-turbo	67.3%	51.8%	GPT-3.5-turbo	100%	64.9%
(a)			(b)		

Table 2 | Results on the R-GSM dataset: (a) accuracies on the full dataset; (b) for each model, the accuracies on the R-GSM subset where the original problems are correctly solved, thus the initial accuracy is 100% for all models.

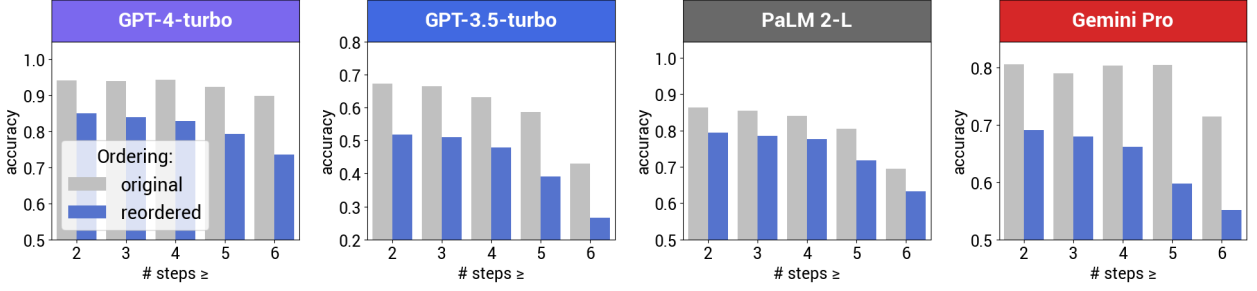


Figure 7 | R-GSM results with different numbers of reasoning steps in the ground truth. See Table 12 in Appendix F for accuracy numbers.

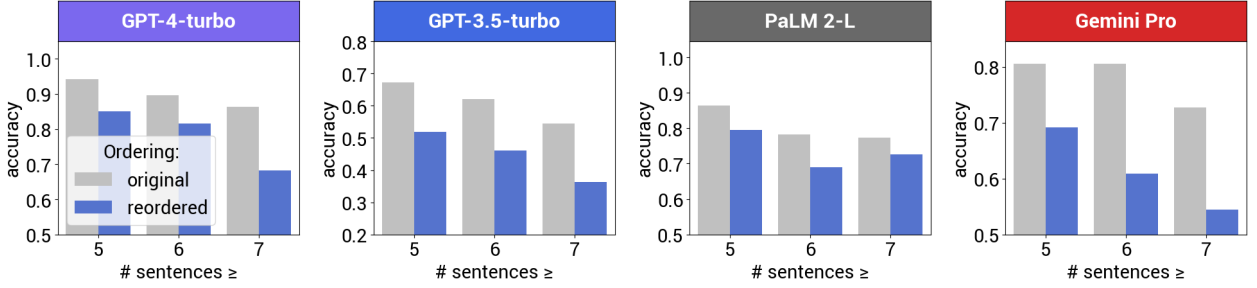


Figure 8 | R-GSM results with different problem lengths. See Table 13 for accuracy numbers.

Table 2a demonstrates the overall results on R-GSM. Again, all LLMs achieve a lower performance on R-GSM. Note that the original GSM8K problems are not necessarily written in the most preferable way, and thus sometimes the manual rewriting facilitates the reasoning and allows the model to correctly solve the reordered version of a problem that it fails on the original one. Therefore, in Table 2b, for each LLM, we also present the accuracy on those problems with their original descriptions solved by the model. We show that all LLMs fail on at least 10% of reordered problems that they are initially able to solve, and this performance degradation is more than 35% with GPT-3.5-turbo.

Breakdown of problem complexity. Figures 7 and 8 present the breakdown results on different number of reasoning steps and different number of problem sentences, respectively. Unsurprisingly, across all LLMs, the proof accuracy suffers on problems that require more reasoning steps and contain a greater number of sentences. Overall, the gap between the accuracies on initial and rewritten problems is more significant with more reasoning steps and longer problems for both GPT-4-turbo and Gemini 1.0 Pro, while the gap remains similar across different numbers of reasoning steps and problem lengths for PaLM 2-L and GPT-3.5-turbo.

Error analysis. To further understand the failure modes, for each LLM, we analyze those error cases where the original problems can be correctly solved but not the reordered ones, and we categorize the common error types in Table 3. Similar to our observation in logical reasoning experiments, the prediction errors in R-GSM are primarily due to the LLMs blindly using numbers in the sequential order of their appearances in the problem. Specifically, the most common error case for all LLMs is their tendency to overlook temporal order. Figure 2 presents such an example, where the prediction failure is because some earlier events are described in the later part of the problem. Another category of errors occurs when some quantities are not specified while processing the problem in the sequential order, which introduces unknown variables for calculation. Take, for example, the problem in Figure 9. In the original problem, the number of each animal can be directly calculated based on its preceding sentence. However, in the reordered problem, the number of gerbils cannot directly be computed

	Temporal	Unknown	Others
GPT-4-turbo	45.0%	15.0%	40.0%
GPT-3.5-turbo	21.6%	19.6%	58.8%
PaLM 2-L	34.8%	4.3%	60.9%
Gemini 1.0 Pro	29.5%	18.2%	52.3%

Table 3 | Error analysis on R-GSM. “Temporal” refers to the temporal order, and “Unknown” refers to the unknown variables.

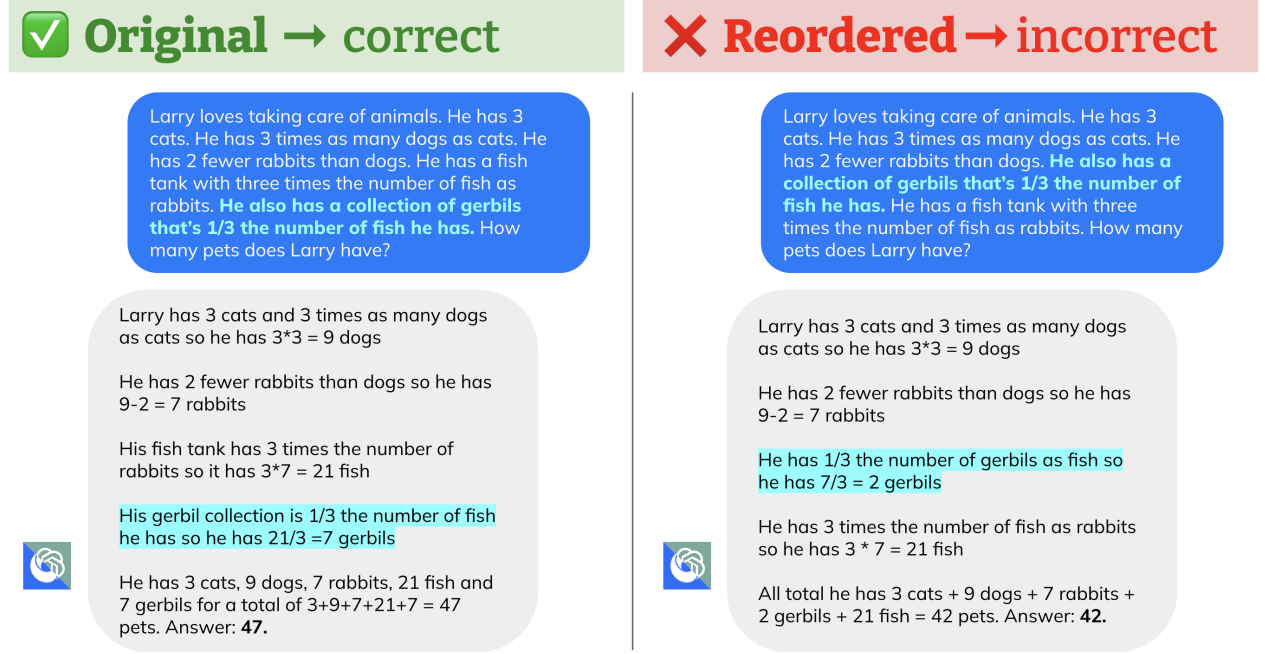


Figure 9 | R-GSM example where the original problem can be correctly solved by all LLMs, but GPT-3.5-Turbo fails on the reordered version while all the other LLMs still solve it correctly.

based on the preceding sentences, since the number of fish remains unknown up to that point, and the LLM must read the remaining sentences and calculate the number of fish first. However, the prediction from GPT-3.5-turbo instead uses the number calculated in the previous step (i.e., the number of rabbits) to calculate the number of gerbils, resulting in an error. Such a failure mode is less common with PaLM 2-L, but still constitutes a non-negligible proportion of prediction errors for the other LLMs. We present more examples of model predictions in Appendix C.

4. Related Work

Failure modes of LLMs. The premise order effect in this work is connected to several failure modes of LLMs in the literature, including the reversal curse (Berglund et al., 2023), distractibility (Shi et al., 2023), position bias (Liu et al., 2024; Wang et al., 2023), and limited capability of logical reasoning (Han et al., 2022; Saparov and He, 2022; Saparov et al., 2023; Wan et al., 2024; Xu et al., 2023; Yan et al., 2023; Zhu et al., 2023). Specifically, Shi et al. (2023) show that including irrelevant context in the problem statement leads to a considerable performance drop on GSM8K and other reasoning benchmarks, revealing that LLMs are *distractible*. This finding is in-line with our

evaluation on logical reasoning, where we observe that adding irrelevant rules not only degrades the overall logical reasoning performance, but also escalates the premise order effect. The *Reversal Curse* (Berglund et al., 2023) unveils another perspective of the order effect, where they show that an LLM that recognizes “A is B” does not necessarily learn that “B is A.” While their work studies the order effect between two entities within a single factual statement, our work focuses on reasoning problems with multiple premises, without restrictions on the number of (or relationship between) entities. In particular, for logical reasoning, we demonstrate that random permutations of premises often result in **worse** accuracy than the purely backward order. Liu et al. (2024) discover the lost-in-the-middle phenomenon in the long-context scenario: the LLM performance is the best when the relevant information to solve the task is placed at the beginning or the end of the input context, while the performance is the worst when the LLM needs to utilize input context in the middle. In Appendix D, we show that lost-in-the-middle phenomenon does not affect the performance on our tasks, since the length of input problems does not exceed 300 tokens in our benchmark, which is relatively small compared to the context length limit of LLMs in our evaluation. Yan et al. (2023) present an approach called Concise and Organized Perception for deductive reasoning, which first generates directed graphs by connecting facts and rules in the problem, then prune and reorder the context accordingly before calling the LLM to solve the problem. The improvement achieved by this approach again demonstrates the effect of premise ordering and irrelevant premises on logical reasoning. While such input preprocessing methods can mitigate the ordering effect on certain reasoning tasks, they require task-specific design and do not generalize across domains. We consider developing generic end-to-end reasoning techniques for LLMs to address the premise order effect as future work.

Order effect for human logical reasoning. Although the premise order does not matter in deductive reasoning, several studies show that the premise order can impact the human reasoning performance (Dekeyser et al., 2000; Girotto et al., 1997). Dekeyser et al. (2000) described *co-reference* as a human preference of premise order; i.e., humans prefer the premises to be presented in an order where they can draw immediate conclusions after seeing each one. In this work, we show that LLMs also have such a preference, and they achieve the best performance when the ordering of rules follows the ground truth proof. Girotto et al. (1997) studied how the premise order affects logical reasoning for humans, and found that the premise order has a significant effect in solving *modus tollens* problems (i.e., if P, then Q; not Q; therefore, not P), but not *modus ponens* problems (i.e., if P, then Q; P; therefore, Q). However, differing from our work, they studied the influence of different ordering between rules and facts, e.g., their experiments on *modus tollens* problems show that presenting negation statements (not Q) before rules (if P, then Q) improves the performance over the reverse order. On the other hand, our work focuses on *modus ponens* problems that are easier for both humans and LLMs, and we show that the LLM performance is still quite sensitive to the ordering of the premises.

Order effect of language models. Some prior works show that language models are able to understand permuted texts to some extent, i.e., after a random permutation of words, models usually preserve a reasonable performance (Abdou et al., 2022; Sinha et al., 2020). Moreover, Cao et al. (2023) show that even when a large fraction of words are scrambled, GPT-4 still achieves decent performance on several reasoning benchmarks. In contrast to permuted texts in these works that are typically unnatural and nonsensical, our premise order permutations do not alter the semantic meaning and remain syntactically valid (we manually verify this). Nevertheless, we demonstrate that LLM reasoning performance is highly brittle to the ordering of the premises. For long-digit addition, prior works demonstrate that reversing the input numbers is a key to achieve better length generalization performance (Lee et al., 2023; Zhou et al., 2023, 2024). Specifically, by reversing the input numbers so that the least significant digit is presented first, the Transformer learns a simpler way

of performing addition, where the model only needs to perform computation with the corresponding digits of operands and the carry-on digit at each step, without the need of looking at other digits. This approach enables the Transformer to better perform addition when trained from scratch, which also aligns with our finding: after reversing the input numbers, the premise order (i.e., orders of digits) follows the right ordering of performing long-digit addition, thus enables Transformers to better learn the task.

5. Conclusion

In this work, we show that the premise order significantly affects LLMs’ performance on reasoning tasks, even when the premise order does not change the underlying task itself. Our comprehensive evaluation demonstrates that LLM tendencies resemble human preference w.r.t. premise order, i.e., LLMs achieve the best performance when the premise order follows the intermediate reasoning steps to solve the problem. Conversely, LLMs face difficulties when the reasoning problem requires the model to read the problem description back-and-forth, resulting in a performance drop of over 30%. We further extend the study to mathematical reasoning and present the R-GSM benchmark, and again experimentally confirm the ordering effect.

While humans also have a preference of premise orders for reasoning problems, LLMs are much more susceptible to such ordering effects. We can attempt to ascribe the premise order effect to several candidate factors, such as the auto-regressive model design, training objectives, and training data mixture. However, we leave proposing theoretical explanations of this limitation and developing new techniques towards addressing the premise order effect as future work.

Acknowledgment

We would like to thank Chen Liang and Dale Schuurmans for helpful discussion and feedback.

References

- M. Abdou, V. Ravishankar, A. Kulmizev, and A. Søgaard. 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, 2022.
- J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- L. Berglund, M. Tong, M. Kaufmann, M. Balesni, A. C. Stickland, T. Korbak, and O. Evans. The reversal curse: Lms trained on "a is b" fail to learn "b is a". *arXiv preprint arXiv:2309.12288*, 2023.
- S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- Q. Cao, T. Kojima, Y. Matsuo, and Y. Iwasawa. 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, 2023.
- M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

- V. A. Cicirello. Kendall tau sequence distance: Extending kendall tau from ranks to sequences. *arXiv preprint arXiv:1905.02752*, 2019.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- M. Dekeyser, W. Schroyens, W. Schaeken, O. Spitaels, and G. d’Ydewalle. Preferred premise order in propositional reasoning: Semantic informativeness and co-reference. *Deductive reasoning and strategies*, pages 73–95, 2000.
- D. Ferreira and A. Freitas. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, 2020.
- Gemini. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- V. Girotto, A. Mazzocco, and A. Tasso. The effect of premise order in conditional reasoning: A test of the mental model theory. *Cognition*, 63(1):1–28, 1997.
- Google. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.
- T. Hagendorff, S. Fabi, and M. Kosinski. Human-like intuitive behavior and reasoning biases emerged in large language models but disappeared in chatgpt. *Nature Computational Science*, 3(10):833–838, 2023.
- S. Han, H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, L. Benson, L. Sun, E. Zubova, Y. Qiao, M. Burtell, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.
- D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- G. Irving, C. Szegedy, A. A. Alemi, N. Eén, F. Chollet, and J. Urban. Deepmath-deep sequence models for premise selection. *Advances in neural information processing systems*, 29, 2016.
- E. Jones and J. Steinhardt. Capturing failures of large language models via human cognitive biases. *Advances in Neural Information Processing Systems*, 35:11785–11799, 2022.
- N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.
- Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.
- N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.
- R. T. McCoy, S. Yao, D. Friedman, M. Hardy, and T. L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

- A. Saparov and H. He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.
- A. Saparov, R. Y. Pang, V. Padmakumar, N. Joshi, S. M. Kazemi, N. Kim, and H. He. Testing the general deductive reasoning capacity of large language models using ood examples. *arXiv preprint arXiv:2305.15269*, 2023.
- P. K. Sen. Estimates of the regression coefficient based on kendall’s tau. *Journal of the American statistical association*, 63(324):1379–1389, 1968.
- F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.
- K. Sinha, P. Parthasarathi, J. Pineau, and A. Williams. Unnatural language inference. *arXiv preprint arXiv:2101.00010*, 2020.
- Y. Wan, W. Wang, Y. Yang, Y. Yuan, J.-t. Huang, P. He, W. Jiao, and M. R. Lyu. $A \& b = b \& a$: Triggering logical reasoning failures in large language models. *arXiv preprint arXiv:2401.00757*, 2024.
- M. Wang, Y. Tang, J. Wang, and J. Deng. Premise selection for theorem proving by deep graph embedding. *Advances in neural information processing systems*, 30, 2017.
- P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu, and Z. Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- F. Xu, Q. Lin, J. Han, T. Zhao, J. Liu, and E. Cambria. Are large language models really good logical reasoners? a comprehensive evaluation from deductive, inductive and abductive views. *arXiv preprint arXiv:2306.09841*, 2023.
- S. Yan, C. Shen, J. Liu, and J. Ye. Concise and organized perception facilitates large language models for deductive reasoning. *arXiv preprint arXiv:2310.03309*, 2023.
- H. Zhang, L. H. Li, T. Meng, K.-W. Chang, and G. V. d. Broeck. On the paradox of learning to reason from data. *arXiv preprint arXiv:2205.11502*, 2022.
- H. Zhou, A. Bradley, E. Littwin, N. Razin, O. Saremi, J. Susskind, S. Bengio, and P. Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.
- Y. Zhou, U. Alon, X. Chen, X. Wang, R. Agarwal, and D. Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.
- Z. Zhu, Y. Xue, X. Chen, D. Zhou, J. Tang, D. Schuurmans, and H. Dai. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*, 2023.

A. R-GSM Dataset Statistics

Table 4 presents the statistics of our R-GSM benchmark.

# Steps	# Problems
2	20
3	43
4	65
5	43
6	23
7	15
8	11

(a)

# Sentences	# Problems
5	133
6	65
7	19
8	3

(b)

Table 4 | Statistics of the R-GSM dataset, with 220 problems in total: (a) breakdown on the number of reasoning steps; (b) breakdown on the number of sentences in the questions.

B. Logical Reasoning Examples

Figure 10 presents common classes of errors — hallucinated rules and facts — by LLMs while solving our logical reasoning benchmark.

Figure 11 presents a sample logical reasoning problem with premise orders of different τ values. We can see that the rules become less ordered when the absolute value of τ decreases.

C. R-GSM Examples

In this section, we present more examples of LLM predictions on R-GSM problems.

Figure 12 presents a failure case of a probability problem, which falls into the “Others” category in the error analysis (Table 3). Specifically, in the reordered problem, after the LLM reads the sentence about the scenario with a normal teacher coming in, the LLM immediately attempts to compute the probability that Marcus has to turn in his homework, ignoring that the LLM needs to compute the probability that a normal teacher will come in using the next sentence.

Figure 13 shows another wrong prediction of GPT-4 Turbo, where the error pattern is analogous to rule hallucination in logical reasoning evaluation. Interestingly, when moving the sentence about yellow cars preceding to the sentence about quantities of blue and green cars, GPT-4 Turbo starts to hallucinate the relationship between the number of yellow cars and the number of blue cars, resulting in insufficient information to correctly solve the problem.

Figures 14 and 15 present examples where both the original and reordered problems are correctly solved by LLMs in our evaluation. In both original problems, the succeeding sentences do not strongly



Figure 10 | Examples of hallucinated rules (left) and facts (right) produced by GPT-3.5-Turbo while solving our logical reasoning benchmark.

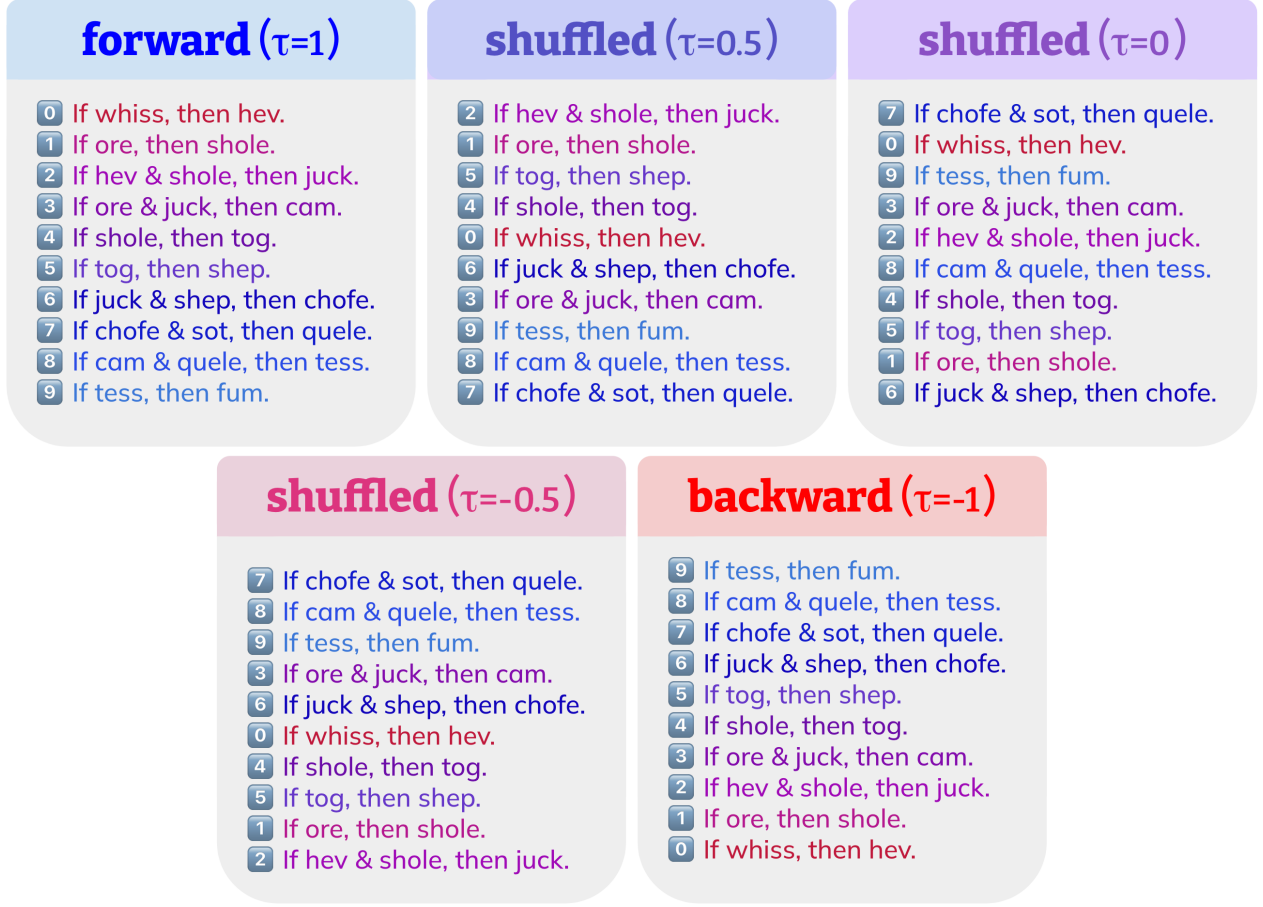


Figure 11 | An example logical reasoning problem with different premise orders. The number emojis are for ease of viewing. The ampersands were originally “and”s in the original prompt. The facts and query have been excluded for brevity.

depend on the preceding sentences.

D. Does Logical Reasoning Suffer from the Lost-in-the-middle Issue?

Liu et al. (2024) demonstrate that when the input context becomes long, LLMs might suffer from the lost-in-the-middle issue: the model performance significantly degrades when relevant information to solve the task is in the middle of the input, instead of at the beginning or the end. Therefore, when given distracting rules for logical reasoning, another potential factor that might affect the model performance is the position of relevant rules in the model input.

To examine the effect of such position bias, we conduct ablations on PaLM 2-L with 10 distracting rules, and we compare the performance with relevant rules added in the beginning, middle or the end of the problem description. Table 5 shows that with the same order and number of rules, the variation in performance is very small, whereas changing the order significantly affects the results. Note that the longest inputs in our logical reasoning benchmark, i.e., problems with 12 relevant rules and 10 distracting rules, only contain no more than 300 tokens, which is relatively short compared to the context length limit of LLMs in our evaluation. These results confirm that on our tasks where the input problems (and thus input context) are short, lost-in-the-middle phenomenon is not the primary cause of the performance difference. In our primary experiments, for all logical reasoning problems,

we interleave distracting rules with relevant rules in the input context.

# rules, position \ order	Forward	Backward	Shuffled
8, beginning	68.0%	40.0%	45.5%
8, middle	67.0%	39.0%	44.5%
8, end	67.0%	40.0%	45.5%
12, beginning	36.5%	17.0%	16.0%
12, middle	36.5%	17.0%	18.5%
12, end	35.0%	16.0%	19.5%

Table 5 | Logical reasoning results performance of PaLM 2-L, with relevant rules at different positions of the input context.

E. Full Results for Logical Reasoning

Tables 6 and 9 present the accuracy numbers for Figures 3 and 5, which are results on different numbers of relevant rules without distracting rules.

Tables 7 and 10 present the accuracy numbers for Figures 4 and 6 with 5 distracting rules.

Tables 8 and 11 present the accuracy numbers for Figures 4 and 6 with 10 distracting rules.

F. Full Results on R-GSM

Tables 12 and 13 present the accuracy numbers for Figures 7 and 8, which are breakdown results on R-GSM problems with different numbers of reasoning steps and different numbers of sentences in the problem description respectively.

✓ **Original → correct**

Marcus is trying to decide whether he really needs to do his homework. **There's a 50% chance that tomorrow he'll have a substitute teacher who won't collect the homework. Even if the normal teacher comes in, there's a 40% chance she'll give everyone an extension.** Even if the whole class doesn't get an extension, there's a 20% chance Marcus can convince the teacher his dog ate his assignment and get a personal extension. What is the percentage chance that Marcus will actually have to turn in his homework tomorrow?

...
Let's break down the events:

1. There's a 50% chance of a substitute teacher who won't collect the homework.
2. If the normal teacher comes in (which has a 50% chance of not happening), there's a 40% chance she'll give an extension to the whole class
3. If there's no class extension (which has a 60% chance of happening if the normal teacher comes in), there's a 20% chance Marcus can get a personal extension by convincing the teacher his dog ate his homework

....
Now, we add up all the probabilities where Marcus does not have to turn in his homework: $P(\text{no homework}) = P(\text{substitute}) + P(\text{normal teacher and extension}) + P(\text{normal teacher, no class extension, and personal extension})$
 $P(\text{no homework}) = 0.50 + 0.20 + 0.06$
 $P(\text{no homework}) = 0.76$ or 76%

Finally, we subtract this from 100% to find the probability that Marcus will have to turn in his homework: $P(\text{homework}) = 1 - P(\text{no homework})$
 $P(\text{homework}) = 1 - 0.76$
 $P(\text{homework}) = 0.24$ or 24%
 So, there is a **24% chance** that Marcus will actually have to turn in his homework tomorrow.



✗ **Reordered → incorrect**

Marcus is trying to decide whether he really needs to do his homework. **If the normal teacher comes in, there's a 40% chance she'll give everyone an extension. There's a 50% chance that tomorrow he'll have a substitute teacher who won't collect the homework.** Even if the whole class doesn't get an extension, there's a 20% chance Marcus can convince the teacher his dog ate his assignment and get a personal extension. What is the percentage chance that Marcus will actually have to turn in his homework tomorrow?

- ...
Let's break it down:
1. The normal teacher comes in and does not give an extension (60% chance of this happening since there's a 40% chance she will give an extension).
 - In this case, Marcus has a 20% chance of getting a personal extension if he claims his dog ate his homework. This means there's an 80% chance he will not get the personal extension and will have to turn in his homework.
 - The probability of this scenario (normal teacher, no class extension, no personal extension) is $60\% * 80\% = 48\%$.
 2. The normal teacher comes in and gives an extension (40% chance of this happening).
 - If the teacher gives an extension, Marcus does not have to turn in his homework tomorrow, so this scenario does not contribute to the probability of Marcus having to turn in his homework.
 3. A substitute teacher comes in (50% chance of this happening).
 - If there's a substitute teacher, Marcus does not have to turn in his homework tomorrow, so this scenario does not contribute to the probability of Marcus having to turn in his homework.

Now, we add up the probabilities of the scenarios where Marcus has to turn in his homework: Probability of having to turn in homework = Probability of scenario 1 = 48%
 So, there is a **48% chance** that Marcus will have to turn in his homework tomorrow.



Figure 12 | R-GSM example where the original problem can be correctly solved by GPT-4 Turbo, but the model fails on the reordered one.

✓ Original → correct

A bumper car rink has 12 red cars. They have 2 fewer green cars than they have red cars. They have 3 times the number of blue cars as they have green cars. **The rink also has yellow cars.** If the rink has 75 cars in total how many yellow cars do they have?

To find out how many yellow cars there are, let's first calculate the number of cars of each other color.

- Red cars: 12

- Green cars: $12 - 2 = 10$

- Blue cars: 3 times the number of green cars
 $= 3 * 10 = 30$

Now, let's add up the red, green, and blue cars to see how many cars we have accounted for:

- Total (Red + Green + Blue) = $12 + 10 + 30 = 52$

The rink has 75 cars in total, so to find out how many yellow cars there are, we subtract the total of red, green, and blue cars from the overall total:

- Yellow cars =
 Total cars - (Red + Green + Blue cars)

- Yellow cars
 $= 75 - 52 = 23$

Therefore, the rink has 23 yellow cars.



✗ Reordered → incorrect

A bumper car rink has 12 red cars. They have 2 fewer green cars than they have red cars. **The rink also has yellow cars.** They have 3 times the number of blue cars as they have green cars. If the rink has 75 cars in total how many yellow cars do they have?

To find out how many yellow cars there are, let's break down the information given:

1. There are 12 red cars.
 2. There are 2 fewer green cars than red cars, so there are $12 - 2 = 10$ green cars.
 3. The number of yellow cars is 3 times the number of blue cars, but we don't know the number of blue cars directly. Instead, we know the relationship between the number of green cars and the number of yellow cars, which seems to have been misstated. Let's correct that understanding: If there's a relationship involving the number of yellow cars directly given, it's not clearly stated in terms of blue or green cars. So, we'll proceed with the information we have.

4. The total number of cars is 75.
 First, let's calculate the total number of cars that are not yellow, since we know the numbers of red and green cars:

Total non-yellow cars = Number of red cars + Number of green cars
 Total non-yellow cars = $12 + 10 = 22$

Given that the total number of cars is 75, we can find the number of yellow cars by subtracting the total number of non-yellow cars from the total:

Number of yellow cars = Total number of cars - Total non-yellow cars
 Number of yellow cars = $75 - 22 = 53$

Therefore, the rink has **53 yellow cars**.



Figure 13 | R-GSM example where the original problem can be correctly solved by all models, but GPT-4 Turbo and Gemini Pro failed on the reordered one.

✓ **Original → correct**

Sheila charged \$85.00 worth of merchandise on her credit card. **She ended up returning one item that cost \$15.00.** After she returned **the** item, she bought a frying pan that was on sale for 20% off \$20.00 and a set of towels that was 10% off \$30.00. She put both of these purchases on her credit card. What is the new balance on her credit card?

✓ **Reordered → correct**

Sheila charged \$85.00 worth of merchandise on her credit card. After she returned **an** item, she bought a frying pan that was on sale for 20% off \$20.00 and a set of towels that was 10% off \$30.00. She put both of these purchases on her credit card. **The returned item cost \$15.00.** What is the new balance on her credit card?



The frying pan was on sale for 20% off \$20.00 so that's $20 \times 0.01 = \$\ll 20 \times 0.01 = 0.20 \gg 0.20$ off

The frying pan was \$20.00 and it's 20% off so that's $20 \times 0.20 = \$\ll 20 \times 0.20 = 4.00 \gg 4.00$ off

The frying pan is \$20.00 and it's \$4.00 off so it now costs $20 - 4 = \$\ll 20 - 4 = 16.00 \gg 16.00$

The towels were 10% off \$30.00 so that's $10 \times 0.01 = \$\ll 10 \times 0.01 = 0.10 \gg 0.10$ off

The towels were \$30.00 and they're 10% off so that's $30 \times 0.10 = \$\ll 30 \times 0.10 = 3.00 \gg 3.00$ off

The towels are \$30.00 and they're \$3.00 off so they now cost $30 - 3 = \$\ll 30 - 3 = 27.00 \gg 27.00$

The frying pan is \$16.00 and the towels are \$27.00 so together they cost $16 + 27 = \$\ll 16 + 27 = 43.00 \gg 43.00$

Sheila charged \$85.00 worth of merchandise and returned an item for \$15.00 so her balance was $85 - 15 = \$\ll 85 - 15 = 70.00 \gg 70.00$

Sheila charged \$43.00 worth of merchandise on her credit card so her new balance is $70 + 43 = \$\ll 70 + 43 = 113.00 \gg 113.00$. **Answer: 113.**

Figure 14 | R-GSM example where both the original and the reordered problems were correctly solved by all LLMs in our evaluation.

✓ **Original → correct**

Tony is painting a room with four walls. The north and south walls are 10 x 8 feet. The east and west walls are 5 x 8 feet. **A gallon of paint can cover 20 square feet and cost \$12.** How much will it cost to paint the room?

✓ **Reordered → correct**

A gallon of paint can cover 20 square feet and cost \$12. Tony is painting a room with four walls. The east and west walls are 5 x 8 feet. The north and south walls are 10 x 8 feet. How much will it cost to paint the room?

First, let's calculate the total area of the walls that Tony is going to paint. The east and west walls are both 5 feet wide and 8 feet tall. So for each of these walls: $\text{Area} = \text{Width} \times \text{Height} = 5 \text{ feet} \times 8 \text{ feet} = 40 \text{ square feet}$

Since there are two of these walls, we need to double that: $\text{Total area for east and west walls} = 2 \times 40 \text{ square feet} = 80 \text{ square feet}$

The north and south walls are both 10 feet wide and 8 feet tall. So for each of these walls: $\text{Area} = \text{Width} \times \text{Height} = 10 \text{ feet} \times 8 \text{ feet} = 80 \text{ square feet}$

Again, since there are two of these walls, we need to double that:
 $\text{Total area for north and south walls} = 2 \times 80 \text{ square feet} = 160 \text{ square feet}$

Now, let's add the total areas of all four walls to get the total area to be painted:
 $\text{Total area} = \text{Area of east and west walls} + \text{Area of north and south walls}$
 $\text{Total area} = 80 \text{ square feet} + 160 \text{ square feet} = 240 \text{ square feet}$

Now we know that one gallon of paint covers 20 square feet and costs \$12. To find out how many gallons Tony needs, we divide the total area by the coverage of one gallon:
 $\text{Gallons needed} = \text{Total area} / \text{Coverage per gallon}$
 $\text{Gallons needed} = 240 \text{ square feet} / 20 \text{ square feet per gallon}$
 $\text{Gallons needed} = 12 \text{ gallons}$

Since paint is typically sold by the gallon, Tony will need to buy 12 gallons of paint to cover the 240 square feet of wall space.

Now, let's calculate the total cost for the paint:
 $\text{Total cost} = \text{Number of gallons} \times \text{Cost per gallon}$
 $\text{Total cost} = 12 \text{ gallons} \times \12 per gallon
 $\text{Total cost} = \$144$

Therefore, it will cost Tony **\$144** to buy the paint needed to cover the walls of the room.



Figure 15 | R-GSM example where both the original and the reordered problems were correctly solved by all LLMs in our evaluation.

# Rules	Order	Acc
4	Forward	99.0%
	Backward	99.5%
	Shuffled	98.8%
5	Forward	98.5%
	Backward	99.5%
	Shuffled	98.2%
6	Forward	100%
	Backward	100%
	Shuffled	98.3%
7	Forward	99.0%
	Backward	98.0%
	Shuffled	97.0%
8	Forward	99.0%
	Backward	95.5%
	Shuffled	93.5%
9	Forward	98.5%
	Backward	95.5%
	Shuffled	93.5%
10	Forward	99.0%
	Backward	92.5%
	Shuffled	87.3%
11	Forward	98.5%
	Backward	91.0%
	Shuffled	87.5%
12	Forward	96.5%
	Backward	84.0%
	Shuffled	80.8%

(a) GPT-4-turbo.

# Rules	Order	Acc
4	Forward	93.0%
	Backward	73.5%
	Shuffled	77.0%
5	Forward	90.0%
	Backward	58.0%
	Shuffled	57.0%
6	Forward	87.5%
	Backward	77.5%
	Shuffled	72.0%
7	Forward	65.5%
	Backward	25.0%
	Shuffled	22.5%
8	Forward	50.0%
	Backward	17.5%
	Shuffled	12.5%
9	Forward	47.5%
	Backward	11.5%
	Shuffled	8.7%
10	Forward	34.0%
	Backward	4.5%
	Shuffled	2.5%
11	Forward	33.0%
	Backward	2.0%
	Shuffled	1.5%
12	Forward	16.5%
	Backward	0.5%
	Shuffled	0.2%

(c) Gemini 1.0 Pro.

# Rules	Order	Acc
4	Forward	98.5%
	Backward	98.5%
	Shuffled	98.3%
5	Forward	98.5%
	Backward	98.5%
	Shuffled	98.3%
6	Forward	98.0%
	Backward	93.5%
	Shuffled	95.3%
7	Forward	96.5%
	Backward	89.0%
	Shuffled	91.2%
8	Forward	95.5%
	Backward	77.0%
	Shuffled	87.7%
9	Forward	94.0%
	Backward	79.0%
	Shuffled	85.7%
10	Forward	95.0%
	Backward	75.5%
	Shuffled	81.0%
11	Forward	94.0%
	Backward	66.0%
	Shuffled	78.7%
12	Forward	88.0%
	Backward	57.5%
	Shuffled	66.5%

(b) PaLM 2-L.

# Rules	Order	Acc
4	Forward	88.5%
	Backward	70.0%
	Shuffled	71.8%
5	Forward	84.0%
	Backward	55.0%
	Shuffled	51.7%
6	Forward	87.5%
	Backward	67.0%
	Shuffled	62.0%
7	Forward	64.0%
	Backward	23.0%
	Shuffled	20.2%
8	Forward	56.5%
	Backward	15.5%
	Shuffled	13.0%
9	Forward	50.5%
	Backward	9.5%
	Shuffled	8.7%
10	Forward	37.0%
	Backward	3.5%
	Shuffled	3.5%
11	Forward	36.0%
	Backward	1.0%
	Shuffled	2.8%
12	Forward	30.0%
	Backward	1.0%
	Shuffled	1.2%

(d) GPT-3.5-turbo.

Table 6 | Result table corresponding to Figure 3.

# Rules	Order	Acc
4	Forward	98.0%
	Backward	99.5%
	Shuffled	99.0%
5	Forward	99.5%
	Backward	98.5%
	Shuffled	98.0%
6	Forward	97.5%
	Backward	97.0%
	Shuffled	96.7%
7	Forward	93.5%
	Backward	92.0%
	Shuffled	90.2%
8	Forward	89.5%
	Backward	85.5%
	Shuffled	82.2%
9	Forward	88.0%
	Backward	84.0%
	Shuffled	82.7%
10	Forward	89.0%
	Backward	77.0%
	Shuffled	74.2%
11	Forward	84.5%
	Backward	75.5%
	Shuffled	71.5%
12	Forward	80.5%
	Backward	72.5%
	Shuffled	57.2%

(a) GPT-4-turbo.

# Rules	Order	Acc
4	Forward	98.5%
	Backward	95.5%
	Shuffled	94.5%
5	Forward	97.0%
	Backward	93.5%
	Shuffled	94.8%
6	Forward	88.0%
	Backward	85.0%
	Shuffled	88.5%
7	Forward	87.5%
	Backward	68.0%
	Shuffled	75.8%
8	Forward	84.5%
	Backward	63.0%
	Shuffled	66.0%
9	Forward	81.5%
	Backward	56.5%
	Shuffled	60.8%
10	Forward	79.5%
	Backward	46.5%
	Shuffled	55.5%
11	Forward	73.0%
	Backward	43.5%
	Shuffled	42.5%
12	Forward	64.0%
	Backward	32.5%
	Shuffled	38.2%

(b) PaLM 2-L.

Table 7 | Results corresponding to Figure 4 with 5 distracting rules.

# Rules	Order	Acc
4	Forward	97.0%
	Backward	98.0%
	Shuffled	97.7%
5	Forward	98.0%
	Backward	96.0%
	Shuffled	96.5%
6	Forward	92.5%
	Backward	88.5%
	Shuffled	90.3%
7	Forward	84.5%
	Backward	80.0%
	Shuffled	76.0%
8	Forward	81.5%
	Backward	76.5%
	Shuffled	70.5%
9	Forward	73.0%
	Backward	65.0%
	Shuffled	62.8%
10	Forward	64.5%
	Backward	59.0%
	Shuffled	53.7%
11	Forward	58.5%
	Backward	53.0%
	Shuffled	48.7%
12	Forward	57.5%
	Backward	46.5%
	Shuffled	40.0%

(a) GPT-4-turbo.

# Rules	Order	Acc
4	Forward	97.5%
	Backward	95.0%
	Shuffled	96.3%
5	Forward	94.0%
	Backward	91.0%
	Shuffled	92.5%
6	Forward	89.0%
	Backward	77.0%
	Shuffled	79.7%
7	Forward	71.5%
	Backward	55.0%
	Shuffled	60.7%
8	Forward	68.5%
	Backward	39.5%
	Shuffled	46.7%
9	Forward	61.5%
	Backward	38.0%
	Shuffled	42.7%
10	Forward	47.0%
	Backward	29.5%
	Shuffled	30.7%
11	Forward	46.5%
	Backward	15.5%
	Shuffled	25.0%
12	Forward	36.5%
	Backward	15.5%
	Shuffled	18.2%

(b) PaLM 2-L.

Table 8 | Results corresponding to Figure 4 with 10 distracting rules.

# Rules	τ	Acc
8	1.0	99.0%
	0.5	95.0%
	0.0	91.0%
	-0.5	94.5%
	-1.0	95.5%
10	1.0	99.0%
	0.5	91.0%
	0.0	82.5%
	-0.5	88.5%
	-1.0	92.5%
11	1.0	98.5%
	0.5	90.0%
	0.0	84.5%
	-0.5	88.0%
	-1.0	91.0%
12	1.0	96.5%
	0.5	76.0%
	0.0	82.0%
	-0.5	84.5%
	-1.0	84.0%

(a) GPT-4-turbo.

# Rules	τ	Acc
6	1.0	87.5%
	0.5	68.5%
	0.0	75.5%
	-0.5	72.0%
	-1.0	77.5%
8	1.0	50.0%
	0.5	10.5%
	0.0	12.0%
	-0.5	15.0%
	-1.0	17.5%
10	1.0	34.0%
	0.5	2.0%
	0.0	3.5%
	-0.5	2.0%
	-1.0	4.5%
12	1.0	16.5%
	0.5	0.0%
	0.0	0.0%
	-0.5	0.5%
	-1.0	0.5%

(c) Gemini 1.0 Pro.

# Rules	τ	Acc
8	1.0	95.5%
	0.5	89.5%
	0.0	86.5%
	-0.5	87.0%
	-1.0	77.0%
10	1.0	95.0%
	0.5	84.0%
	0.0	83.0%
	-0.5	76.0%
	-1.0	75.5%
11	1.0	94.0%
	0.5	80.5%
	0.0	76.5%
	-0.5	79.0%
	-1.0	66.0%
12	1.0	88.0%
	0.5	74.5%
	0.0	65.5%
	-0.5	59.5%
	-1.0	57.5%

(b) PaLM 2-L.

# Rules	τ	Acc
6	1.0	87.5%
	0.5	68.5%
	0.0	75.5%
	-0.5	72.0%
	-1.0	77.5%
8	1.0	50.0%
	0.5	10.5%
	0.0	12.0%
	-0.5	15.0%
	-1.0	17.5%
10	1.0	34.0%
	0.5	2.0%
	0.0	3.5%
	-0.5	2.0%
	-1.0	4.5%
12	1.0	16.5%
	0.5	0.0%
	0.0	0.0%
	-0.5	0.5%
	-1.0	0.5%

(d) GPT-3.5-turbo.

Table 9 | Result table corresponding to Figure 5.

# Rules	τ	Acc	# Rules	τ	Acc
8	1.0	89.5%	8	1.0	84.5%
	0.5	86.5%		0.5	67.5%
	0.0	78.0%		0.0	67.0%
	-0.5	82.0%		-0.5	63.5%
	-1.0	85.5%		-1.0	63.0%
10	1.0	89.0%	10	1.0	79.5%
	0.5	75.5%		0.5	58.0%
	0.0	70.5%		0.0	56.0%
	-0.5	76.5%		-0.5	52.5%
	-1.0	77.0%		-1.0	46.5%
11	1.0	84.5%	11	1.0	73.0%
	0.5	68.5%		0.5	41.5%
	0.0	67.5%		0.0	40.0%
	-0.5	78.5%		-0.5	46.0%
	-1.0	75.5%		-1.0	43.5%
12	1.0	80.5%	12	1.0	64.0%
	0.5	49.5%		0.5	39.0%
	0.0	61.5%		0.0	42.0%
	-0.5	60.5%		-0.5	33.5%
	-1.0	72.5%		-1.0	32.5%

(a) GPT-4-turbo.
(b) PaLM 2-L.

Table 10 | Results corresponding to Figure 6 with 5 distracting rules.

# Rules	τ	Acc
8	1.0	81.5%
	0.5	73.0%
	0.0	65.5%
	-0.5	73.0%
	-1.0	76.5%
10	1.0	64.5%
	0.5	48.5%
	0.0	50.5%
	-0.5	62.0%
	-1.0	59.0%
11	1.0	58.5%
	0.5	54.0%
	0.0	41.0%
	-0.5	51.0%
	-1.0	53.0%
12	1.0	57.5%
	0.5	33.0%
	0.0	42.0%
	-0.5	45.0%
	-1.0	46.5%

(a) GPT-4-turbo.

# Rules	τ	Acc
8	1.0	68.5%
	0.5	48.5%
	0.0	45.5%
	-0.5	46.0%
	-1.0	39.5%
10	1.0	47.0%
	0.5	35.0%
	0.0	30.0%
	-0.5	27.0%
	-1.0	29.5%
11	1.0	46.5%
	0.5	30.0%
	0.0	24.5%
	-0.5	20.5%
	-1.0	15.5%
12	1.0	36.5%
	0.5	18.0%
	0.0	19.0%
	-0.5	17.5%
	-1.0	15.5%

(b) PaLM 2-L.

Table 11 | Results corresponding to Figure 6 with 10 distracting rules.

# Steps	Init Acc	Reorder Acc
≥ 2	94.1%	85.0%
≥ 3	94.0%	84.0%
≥ 4	94.3%	82.8%
≥ 5	92.4%	79.3%
≥ 6	89.8%	73.5%

(a) GPT-4-turbo.

# Steps	Init Acc	Reorder Acc
≥ 2	86.4%	79.5%
≥ 3	85.5%	78.5%
≥ 4	84.1%	77.7%
≥ 5	80.4%	71.7%
≥ 6	69.4%	63.3%

(b) PaLM 2-L.

# Steps	Init Acc	Reorder Acc
≥ 2	80.5%	69.1%
≥ 3	79.0%	68.0%
≥ 4	80.3%	66.2%
≥ 5	80.4%	59.8%
≥ 6	71.4%	55.1%

(c) Gemini 1.0 Pro.

# Steps	Init Acc	Reorder Acc
≥ 2	67.3%	51.8%
≥ 3	66.5%	51.0%
≥ 4	63.1%	47.8%
≥ 5	58.7%	39.1%
≥ 6	42.9%	26.5%

(d) GPT-3.5-turbo.

Table 12 | Results corresponding to Figure 7.

# Sentences	Init Acc	Reorder Acc	# Sentences	Init Acc	Reorder Acc
≥ 5	94.1%	85.0%	≥ 5	86.4%	79.5%
≥ 6	89.7%	81.6%	≥ 6	78.2%	69.0%
≥ 7	86.4%	68.2%	≥ 7	77.3%	72.7%
(a) GPT-4-turbo.			(b) PaLM 2-L.		
# Sentences	Init Acc	Reorder Acc	# Sentences	Init Acc	Reorder Acc
≥ 5	80.5%	69.1%	≥ 5	67.3%	51.8%
≥ 6	80.5%	60.9%	≥ 6	62.1%	46.0%
≥ 7	72.7%	54.5%	≥ 7	54.5%	36.4%
(c) Gemini 1.0 Pro.			(d) GPT-3.5-turbo.		

Table 13 | Results corresponding to Figure 8.