The Curse of Verbalization: How Presentation Order Constrains LLM Reasoning

Anonymous ACL submission

Abstract

This paper delves into the factors that contribute to the difficulty of problems for large language models (LLMs). We begin with a pilot test evaluating LLMs' understanding of esoteric programming languages and find that LLMs struggle significantly when programs execute in an order that is unaligned with how the program is presented. This phenomenon leads to the hypothesis that LLM performance on reasoning correlates with the alignment between the order in which information is presented and the order in which it should be utilized. We demonstrate that this hypothesis holds broadly in mathematical reasoning: restructuring problems to align the order of information presentation with the order of utilization consistently improves performance across state-of-the-art models. We conjecture this occurs because LLMs acquire a strong tendency to verbalize information in presentation order during training on human text, a tendency detrimental in reasoning domains where the optimal utilization order often diverges from the presentation order. To provide further evidence, we construct pseudo-mathematical problems with nonsensical terms and quantify the verbalization flexibility of LLMs without interference from mathematical knowledge. Across twelve representative LLMs, we find that this flexibility exhibits a strong correlation ($\rho = 0.87$) with general reasoning performance rankings on LMArena.

1 Introduction

The relentless progress of large language models' (LLMs) "reasoning" abilities has elicited the development of increasingly challenging benchmarks for their evaluation (Cao et al., 2025; Xu et al., 2025; Besta et al., 2025). Yet while the *difficulty* of a problem can be readily quantified by measuring an LLM's performance, understanding *why* specific problems pose a challenge to the LLM remains considerably more complex: First, LLMs' failure can be non-intuitive; they can successfully solve complex problems while struggling with those considered trivial for humans (Song et al., 2025; Huang and Chang, 2023). Furthermore, performance is often model-dependent and further complicated by benchmark contamination, meaning the solvability of a problem may stem from data memorization rather than being genuinely easy for the LLM (Xu et al., 2024; Dong et al., 2024). Lastly, the high dimensionality of a problem, involving numerous semantic features and variables, hinders the isolation of the factor causing a model's failure (Singh et al., 2024; Qi et al., 2025).

Thus, to facilitate the study of factors that contribute to the difficulty of a problem for LLMs, we may start by seeking tasks that satisfy several key criteria. First, the task should be minimally represented in LLM pre-training data and less aligned with typical post-training objectives. Second, the task presentation should be simple and direct, with more symbolic structure to minimize confounding factors.

We find that certain esoteric programming languages (Esolangs) are ideal candidates that satisfy these principles. We begin with a pilot study examining LLM performance on Befunge, an esolang that employs directional operators (<, >, ^, v) to control program counter movement on a 2-D grid. It introduces a distinct feature regarding 'information sequencing': to evaluate a Befunge program's output, one must traverse and utilize cell information not necessarily from left to right, but in varying orders determined by the directional operators encountered during execution.

Our experiments reveal that the LLM understands the program substantially better when the execution proceeds from left to right (in the same order as the program tokens input to the LLM), but struggles when programs require backward traversals. These observations lead to a hypothesis: LLM performance on reasoning correlates with the align-

ment between the order in which information is presented and the order in which it should be utilized. When the two orders are aligned, problems become easier for LLMs to solve.

But is this phenomenon specific to Befunge, or does it reflect a more fundamental characteristic of LLM reasoning? To test this hypothesis, we extend our investigation to the domain of mathematical reasoning. Specifically, we investigate whether the solvability of a problem can be improved by aligning the presentation order with the utilization order, that is, by restructuring problem statements so that information appears in the sequence it is processed during solution. Our evaluation across GSM8K (Cobbe et al., 2021a), MATH (Hendrycks et al.), and AIME using both general-purpose and reasoning LLMs (GPT-3.5-turbo, GPT-4o (OpenAI, 2024), Claude 3.7 (Anthropic, 2025), DeepSeek-R1 (Guo et al., 2025)) shows that reordered problems with presentation order aligned to utilization order consistently improve both pass@k and major@k performance, while deliberately reversing that presentation order mostly degrades performance.

Why does this phenomenon occur? We argue that the "reasoning" process, or more broadly, the process by which LLMs answer arbitrary questions, is fundamentally a process of verbalization: the model must select an order in which to verbalize its output, sequentially connecting critical pieces of input information. Through pre-training on human data and subsequent alignment, LLMs acquire a strong tendency to verbalize information in the order it is presented. While this tendency is so natural and probably does no harm for numerous tasks, it becomes problematic in domains such as mathematics and programming, where the optimal utilization order often diverges from the presentation order. We term this bias, which constrains the model's reasoning, the **Curse of Verbalization**.

To find more evidence for this hypothesis, we constructed pseudo-mathematical questions with nonsensical terms, allowing us to directly observe how LLMs choose to verbalize through these elements without interference with pre-trained mathematical knowledge. We find that weaker LLMs exhibit a noticeable tendency to verbalize these terms in the exact order they appear in the problem statement. In comparison, stronger LLMs display more creative and flexible verbalization patterns. Remarkably, this verbalization flexibility shows a strong correlation (0.87) with overall reasoning performance in coding and math.

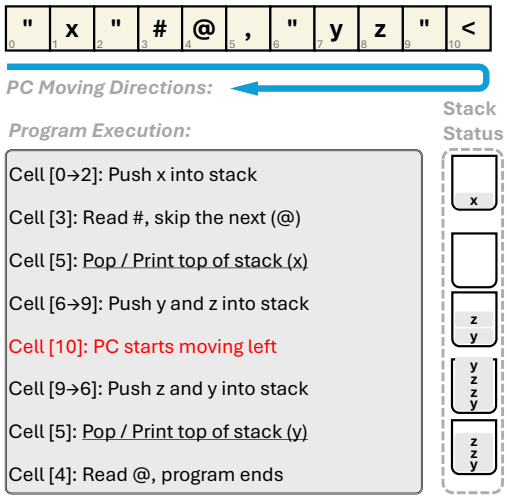

Figure 1: An example of → then ← pattern on 1-D Befunge program. The printed out result is "xy".

Category	Input	Output
← then →	`<>"foo",,@#"bar"`	baoo
← only	`<@,,"bar"`	ba
→ only	`>"foo",,,@`	oof
→ then ←	`>"foo"#@,,"bar"<`	ooba
→ then ← then ←	`>#,"foo"<@,,"bar"`	fba

Table 1: Befunge program execution patterns with input/output examples.

2 A Pilot Test with Befunge

An esoteric programming language (Esolang)[1] is a programming language designed to test the boundaries of computer programming language design, often with unique and more symbolic representations of data and instructions. Like mainstream programming languages, understanding esolangs demands reasoning; however, esolangs maintain minimal presence on the internet, and they are likely absent from deliberate alignment processes that shape modern LLMs. This unique position makes esolangs a contamination-resistant testbed for understanding problem difficulties in LLMs.

For a pilot test, we employ Befunge[2], an esolang featuring a two-dimensional grid of cells with directional operators (<, >, ^, v) that control program counter movement. To reduce the problem space, we simply ask the LLM to evaluate a 1-D program that only prints out no more than five characters. The instructions used in the experiments are also limited (See Table 2). Figure 1 demonstrates a simple program and how it is evaluated. If not

[1] https://en.wikipedia.org/wiki/Esoteric_programming_language

[2] https://esolangs.org/wiki/Befunge

Instruction	Description
>	PC starts moving right
<	PC starts moving left
"	Toggle stringmode (push each character's ASCII value into the stack until the next "
,	Pop top of stack and output as ASCII character
@	End the program
#	Bridge; jump over next command in the current direction of the current PC
_	Horizontal If; Pop a value; set direction to right if value = 0, set to left otherwise
:	Duplicate top stack value
\	Swap top stack values

Table 2: Included Befunge Instructions and their meanings.

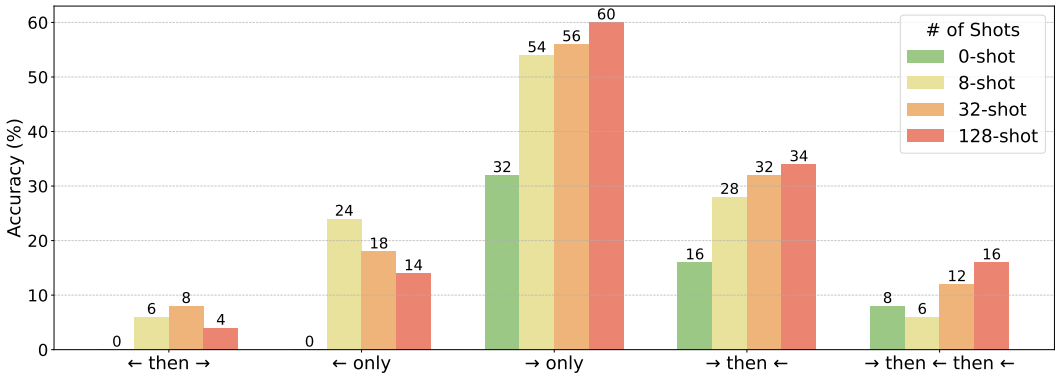

Figure 2: Accuracy of GPT-4o on Befunge program evaluation across different execution patterns and numbers of in-context examples.

specified, the program counter starts from the leftmost cell and moves to the right. As the program reads <, the program counter starts moving to the left and ends when it reads @. In this case, the PC moves to the right, then to the left, which we refer to as the pattern of "→ then ←". Similarly, on a 1-D grid, we can have four other patterns, as shown in Table 1 (e.g., "→ only" refers to programs that execute exclusively from left to right).

We randomly generate 200 such simple programs per pattern using a script and prompt GPT-4o to predict the printed output at both zero-shot and few-shot settings, reporting the accuracy of the evaluation. Results in Figure 2 show that different patterns have distinct difficulties: "→ only" is the easiest, followed by "→ then ←". There is a significant accuracy drop whenever the program executes, starting from the rightmost cell, or involves multiple traversals. As the number of in-context examples increases, performance improves, possibly due to the direct retrieval of similar programs from these examples.

Notably, the design of Befunge introduces a distinct feature regarding 'information sequencing': Every Befunge program implicitly provides the ground truth order of information usage — the precise sequence in which each cell's information must be processed, as dictated by the directional operators. Since the **information presentation order**, i.e., the order of the program tokens input to the LLM, is trivially left-to-right, this **information utilization order** can diverge from the presentation order (e.g., a program beginning with < instructs the program counter to jump to the rightmost cell and traverse leftward).

These observations lead us to propose a hypothesis: *LLM performance on reasoning tasks correlates strongly with the alignment between the order in which information is presented and the order in which it must be utilized.* When the two orders are aligned, problems become easier for LLMs to solve (in the case of Befunge, "→ only" ensures that both information presentation order and utilization order are exactly the same). Conversely, when the information utilization order is misaligned with the presentation order (e.g., "← only"), the problem poses more challenges to an LLM. This hypothesis also suggests that LLMs may possess an inherent bias toward utilizing the critical information in the order it is given, struggling to manipulate alternative (and possibly more optimal) orderings to solve the problem correctly.

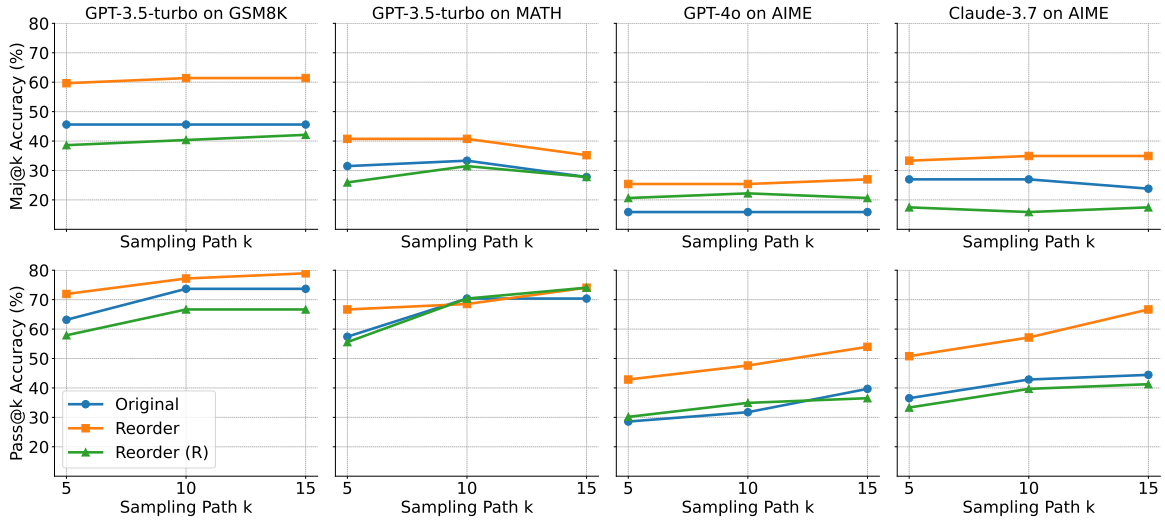

Figure 3: Performance comparison across original, aligned (Reorder), and reversed (Reorder (R)) problem variants. Aligned problems consistently improve both pass@k and major@k accuracy.

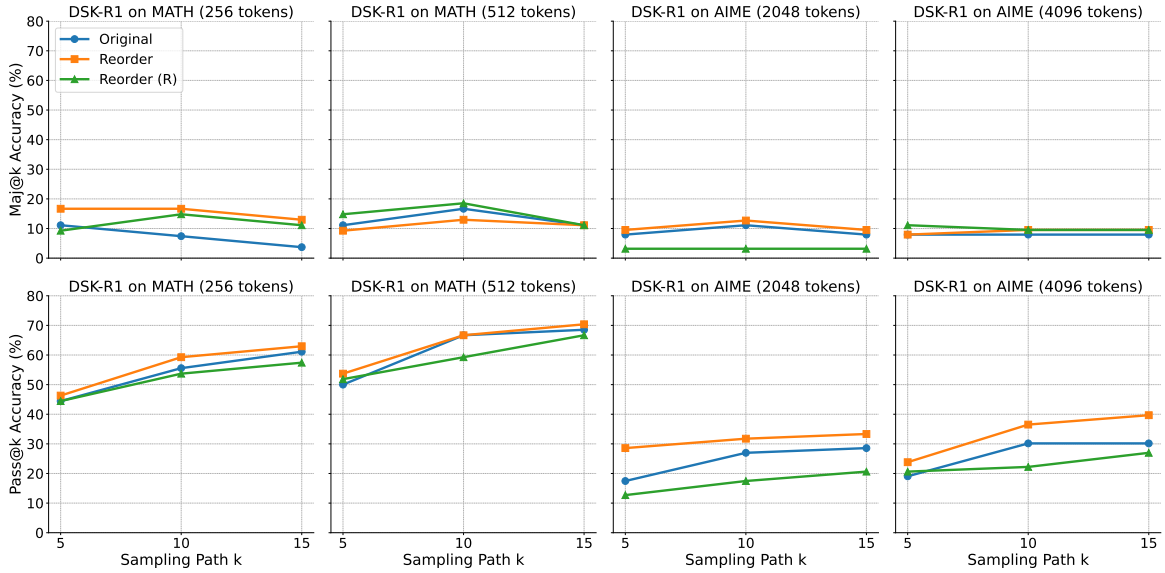

Figure 4: Performance comparison of DeepSeek R1 on MATH and AIME with various budgets.

3 Order Effects in Math Reasoning

While we observe how this misaglinment affects difficulty in Befunge, a critical question remains: Is this phenomenon specific to obscure esolangs, or does it persist in prevailing reasoning tasks where LLMs have been extensively optimized?

In this section, we test whether such order effects persist in mainstream mathematical reasoning tasks. The core idea is to take a set of math problems and create variations where the level of alignment between the presentation order and utilization order is different. Notably, while Befunge provides a clear ground truth reference for information utilization order (determined by the program counter's execution path), no such ground truth exists for math problems. To solve a math problem, there can be multiple valid reasoning paths, each utilizing the given conditions and variables in a different sequence. We therefore use the **ground truth solution's order as a proxy for the optimal utilization order**. Another challenge in distinguishing math problems from Befunge is the absence of distinct data and instructions in discrete cells; we cannot treat each word as a unit of information without making the problem intractable. Instead, we define **critical information** as the conditions, variables, and numerical values in the problem that must be used to construct the solution. For instance, given the problem "Alex writes 2 books a year, how many

books did he write in 5 years?", the critical information consists of "2 books a year" and "5 years", and we focus exclusively on the ordering of these elements.

For each problem, we create two variants through a two-step rewriting process to avoid information leakage from the solution. **First**, we prompt an LLM to extract the critical information from the ground truth solution in the order it appears, producing a sequence of information elements. **Second**, we ask the LLM to rewrite the original problem p such that the presentation order of critical information is either: (1) **aligned** with the solution order, or (2) **completely reversed** relative to the solution order. Crucially, during the paraphrasing step, the LLM receives only the sequence of critical information elements. This process yields two reordered problem sets from the original problems, which we denote as $\mathcal{P}_{\text{Original}}, \mathcal{P}_{\text{Reorder}}, \mathcal{P}_{\text{Reorder (R)}}$.

To compare model performance across these problem sets, we ask the LLM to solve each problem K times. We then compute two complementary metrics: **Pass@k** measures whether *at least one* of the K attempts produces the correct answer, capturing the model's ceiling performance or inherent solvability of the problem (Chen et al., 2021b). **Major@k** measures whether the majority of attempts are correct, capturing solution consistency and reflecting how reliably or easily the model can solve the problem (Wang et al., 2023).

Experimental Settings We employ three mathematical reasoning benchmarks of varying difficulty: **GSM8K** (Cobbe et al., 2021b), a dataset of grade-school level math word problems; **MATH** (Hendrycks et al., 2021), a competition mathematics dataset with challenging concepts such as Calculus and Linear Algebra; and **AIME 2023–2024**[3], consisting of problems from the American Invitational Mathematics Examination, representing the highest difficulty tier. From each dataset, we sample problems where the original critical information presentation order differs from the solution order, and where at least three critical information elements exist. This sampling ensures sufficient complexity to observe presentation-utilization order effects. For AIME, multiple solutions are provided for each problem, and we use the shortest solution for information extraction. We test four LLMs spanning different stages of LLM development, from vanilla general-purpose models to specialized thinking models: **GPT-3.5-turbo**, **GPT-4o** (OpenAI, 2024), **Claude 3.7 Sonnet** (Anthropic, 2025), and **DeepSeek-R1** (Guo et al., 2025). We use GPT-4o as the paraphrasing model across all settings. In general, we pair stronger models with more difficult datasets to ensure meaningful performance variance, as stronger models can easily solve simpler problems. Additionally, we evaluate **DeepSeek-R1** (Guo et al., 2025), a state-of-the-art reasoning model featuring extremely long thinking, under varying computational budgets by imposing different token limits on its generations, allowing us to examine how presentation-utilization order alignment enables the model to solve problems with fewer budget. We choose $K \in \{5, 10, 15\}$ for computing both pass@k and major@k metrics. All experiments use a temperature $T = 0.7$.

Experimental Results Figure 3 shows the performance of four model-dataset pairs regarding pass@k and major@k accuracy at k = 5, 10, 15, across original problems, aligned problems (Reorder), and reversed problems (Reorder (R)). In general, aligned problems achieve higher major@k and pass@k accuracy at all k values, while reversed problems tend to perform worse than the original. In detail, aligned problems consistently improve major@k across all settings, indicating that rearranging critical information in the problem presentation makes problems generally more likely to be solved. Surprisingly, this alignment can also improve pass@k, especially for AIME, demonstrating that certain problems can be potentially solved by the LLM when presentation order is aligned with utilization order. The reversed problems do not degrade accuracy as substantially as aligned problems improve it. In certain settings, including GPT-4o on AIME and GPT-3.5-turbo on MATH, the reversed version performs slightly better than the original, although the improvement is marginal.

Figure 4 shows the performance of DeepSeek-R1 on MATH and AIME under different computational budgets, with lower budgets allocated to the easier dataset (MATH) and higher budgets to AIME. We observe a similar pattern in Pass@k, where $\text{Acc}_{\text{aligned}} > \text{Acc}_{\text{original}} > \text{Acc}_{\text{reverse}}$ across budget levels. However, for Major@k, the differences are only marginal, and the pattern is less clear on MATH. Contrary to our expectation, alignment does not yield greater accuracy improvements at

[3] https://artofproblemsolving.com/wiki/index.php/

low budgets compared to high budgets, suggesting that presentation-utilization order alignment does not reduce the computational cost required to solve problems.

4 Correlation between Verbalization Flexibility and Reasoning Ability

The empirical results across Befunge and mathematical reasoning benchmarks demonstrate that aligning the information presentation order with utilization order facilitates LLM problem-solving performance, while misalignment degrades it. But why does this phenomenon occur?

We argue that the "reasoning" process, or more broadly, the process by which LLMs answer arbitrary questions, is fundamentally a **process of verbalization**: the model must select an order in which to verbalize its output, sequentially connecting critical pieces of input information.

Through daily use of any LLM, one may notice that LLMs tend to generate output by addressing conditions and variables in the order they appear in the input. This may suggest that through pretraining on human data and subsequent alignment, LLMs acquire a strong tendency to verbalize information in the order it is presented. While this tendency is natural and probably does no harm for numerous tasks, it can impede performance in reasoning domains such as mathematics and programming, where the optimal utilization order often diverges from the presentation order. Thus, we hypothesize that there is a negative correlation between this tendency and the general reasoning performance of LLMs, which we term the **Curse of Verbalization**.

To validate this hypothesis, we use the math and coding reasoning rankings from the LMArena leaderboard[4] as the proxy of an LLM's general reasoning ability instead of relying on solvability of any specific dataset. To gauge LLMs' verbalization tendency without entanglement with data contamination or LLMs' pretrained knowledge while still inducing their reasoning process, we create a benchmark containing pseudo-mathematical problems using nonsensical terms. Specifically, we train a Markov generator[5] to generate the skeleton of math problems based on the training data from GSM8K and MATH. We then replace nouns with nonsensical sci-fi terms from Rick and Morty[6] (colored text). For example:

> *"For krombopulos, globaflyn of glem from smidgen to keep for hizzard. chumble is playing plumbus is 0.75. glem of drawing fleeb or chumble is? Think step by step."*

Such superficially math-like problems are entirely nonsensical and unsolvable. This design is intentional and serves multiple purposes. First, the problems retain the structure of multi-step reasoning, thereby inducing LLMs to engage their reasoning processes and formulate a solution strategy. Second, because the terms carry no semantic meaning, any pattern in how models verbalize these terms must reflect their inherent verbalization biases rather than mathematical knowledge or memorized solution templates.

To quantify the verbalization tendency, we extract the order of these sci-fi terms as they appear in both the problem statement and the LLM's output, then compute the **normalized edit distance** between these two orderings:

$$\frac{\text{EditDistance}(O_{\text{problem}}, O_{\text{output}})}{\max(|O_{\text{problem}}|, |O_{\text{output}}|)}$$

A normalized edit distance of 0 indicates perfect adherence to presentation order, while higher values indicate greater *flexibility* in reordering information during verbalization.

We select twelve representative LLMs and for each LLM, we randomly generate 300 pseudo-math questions and compute the averaged normalized edit distance as the overall measure of its **verbalization flexibility (VF)**. We also report each model's ranking in math and coding from the LMArena leaderboard, where lower ranks indicate stronger performance. The results, sorted by verbalization flexibility, are presented in Table 3.

We find that verbalization flexibility exhibits a strong positive correlation with reasoning performance. Models with higher VF values, indicating greater flexibility in reordering information during verbalization, consistently achieve better rankings in both math and coding tasks. We observe a Pearson correlation coefficient of -0.87 and -0.85 between the VF values and coding and math rank, respectively.

[4] https://lmarena.ai/leaderboard
[5] https://pypi.org/project/markovify/
[6] https://en.wikipedia.org/wiki/Rick_and_Morty

Model Card	VF Score	$\text{Rank}_{\text{coding}}$	$\text{Rank}_{\text{math}}$
Qwen3-235B-A22B-Thinking-2507	0.788	3	2
DeepSeek-R1	0.758	18	8
gemini-2.5-flash	0.730	28	7
gemini-1.5-pro	0.659	67	53
Qwen2.5-72B-Instruct	0.636	73	69
Llama-3.3-70B-Instruct	0.590	87	72
Meta-Llama-3.1	0.574	108	97
claude-3-7-sonnet-20250219	0.559	21	18
claude-3-haiku-20240307	0.542	123	129
gpt-4o-mini	0.538	82	90
gpt-3.5-turbo-0125	0.502	143	147
Mixtral-8x7B-Instruct-v0.1	0.425	163	152

Table 3: Verbalization flexibility (VF) strongly correlates with performance ranking in math and coding.

One might argue that thinking models naturally achieve higher edit distances simply because they generate more tokens, potentially mentioning each term multiple times. However, this concern is unfounded for two reasons. First, the edit distance is normalized by sequence length, accounting for differences in output verbosity. Second, and more importantly, the key factor is not output length but reordering behavior. Consider that if s_1 = "abc" and s_2 = "cba", the normalized distance is 0.667, whereas if s_2 = "aabbcc", the distance is only 0.5. This demonstrates that weaker models could achieve high edit distances by even moderately reordering the terms—however, they rarely do so. In fact, if we only examine the rankings of thinking models, we observe that verbalization flexibility remains correlated with reasoning performance even within this subset. Among thinking models, the stronger one tends to be the one with greater flexibility. This suggests that verbalization flexibility is not merely a binary distinction between thinking and non-thinking models, but rather a continuous capability that could scale with reasoning strength across all model types.

Table 4 presents qualitative examples of how different models use the sci-fi terms when answering the example question presented earlier. Vanilla chat LLMs such as GPT-3.5-turbo and Mixtral almost invariably follow whatever order the sci-fi words appear in the problem statement and use them accordingly. In contrast, strong thinking models tend to have distinct preferences and unfold their reasoning around them. Notably, stronger models may never use certain terms at all or discover related however unmentioned terms (e.g., "schleem" in DeepSeek R1) for reasoning, demonstrating a selective and strategic approach to information utilization that weaker models lack.

5 Related Work

The advancement of LLMs' reasoning ability has prompted the development of more challenging benchmarks in math and coding, from the classic GSM8K (Cobbe et al., 2021b), MATH (Hendrycks et al., 2021), and HumanEval (Chen et al., 2021a), to the more challenging ones, such as AIME and LiveCodeBench (Jain et al., 2024). Meanwhile, reasoning frameworks themselves have evolved rapidly, from early chain-of-thought and related sampling-based and stepwise verification approaches (Wei et al., 2022; Yao et al., 2023; Besta et al., 2023; Wang et al., 2023; Huang and Chang, 2023) to recent reward models with test-time computation or RL–driven reasoning optimization (Besta et al., 2025; Xu et al., 2025).

However, as LLMs grow in scale with pretraining data and alignment sophistication, as well as their ability to verbalize seemingly plausible reasoning processes in natural language, it has become increasingly difficult to interpret their outputs, particularly when they fail in certain cases (Xu et al., 2024; Dong et al., 2024; Singh et al., 2024; Song et al., 2025). While recent research has begun to focus on identifying challenging reasoning problems and quantifying their difficulty through various metrics (Shao et al., 2024), comparatively little work has examined the intrinsic characteristics of problems that actually contribute to their difficulty for LLMs (Zhou et al., 2024; Srivatsa and Kochmar, 2024).

6 Discussion and Conclusion

In this paper, we employ contamination-resistant symbolic tasks as an entry point to study problem difficulty for LLMs. Then we demonstrate that our findings generalize to mathematical rea-

Model	Score	Model Solution
GPT-3.5-turbo	0.0	1. Determine the value of krombopulos, globaflyn, and glem from smidgen to keep for hizzard. 2. Calculate the value of chumble playing plumbus at 0.75. 3. Decide if glem of drawing fleeb or chumble is the better option based on the calculated values.
Claude-3.7	0.622	I'll solve this probability problem step by step. Let me define the events: - Let F = drawing a fleeb - Let C = drawing a chumble - We're told that $P(C) = 0.75$ (probability of chumble is 0.75) - We need to find $P(F \cup C)$, which is the probability of drawing either a fleeb or a chumble Using the addition rule of probability: $P(F \cup C) = P(F) + P(C) - P(F \cap C)$ Where: - $P(F \cup C)$ is the probability of drawing fleeb or chumble - $P(F)$ is the probability of drawing fleeb - $P(C)$ is the probability of drawing chumble (given as 0.75) - $P(F \cap C)$ is the probability of drawing both fleeb and chumble However, I don't have enough information to determine $P(F)$ or $P(F \cap C)$ ...
DSK-R1	0.761	...This seems like nonsense, but it might be a reference to something. I recall that "plumbus" and "chumble" are from Rick and Morty. In the show, a plumbus is a common household item, and there's a description of how it's made... Let me think about the context. In Rick and Morty, the plumbus creation process involves several steps: schleem is rubbed on by a chumble, fleeb is used, etc. But here, it says "chumble is playing plumbus is 0.75." That might mean the probability or some value associated with chumble and plumbus is 0.75. In the plumbus episode, there's " schleem ," not "glem." Let me check the user's message: "globaflyn of glem" – that doesn't ring a bell... Let's read carefully: "globaflyn of glem" – maybe "globaflyn" is "probability"? That could be it. "Probability of glem" or something. And "from smidgen to keep for hizzard" – not sure. "smidgen" might be a small amount, but in context, it could be a state or something...But what is "glem"? ...
Gemini-2.5-flash	0.711	To answer your question about the glem of drawing fleeb or chumble, I would need: - A definition for "glem": Assuming "glem" means probability, which is the most likely interpretation in a problem with a given decimal value like 0.75. - The specific context and relationship between "chumble" and "plumbus": You are given that the "glem" (probability) that chumble is playing plumbus is 0.75. This likely means: $$P(chumble) = 0.75$$ Where $P(chumble)$ is the probability of the event "chumble" occurring. - The "glem" (probability) for "fleeb": The probability of drawing a fleeb, $P(fleeb)$, is missing. - The relationship between "fleeb" and "chumble": To find the probability of "fleeb or chumble," I need to know if these events are mutually exclusive (cannot happen at the same time) or overlapping...

Table 4: Qualitative Examples of how different LLMs utilize sci-fi terms during verbalization.

soning. We demonstrate that problems become easier when the information presentation order aligns with the utilization order, and more difficult when it is misaligned. Exploring the mechanism behind this phenomenon, we identify the Curse of Verbalization: LLMs develop a strong tendency to process and verbalize information in presentation order, which is negatively related to their reasoning capabilities. We believe this pipeline offers a principled approach to studying problem difficulty: first, isolate factors in contamination-resistant symbolic tasks, then validate their generalization to mainstream domains. By examining factors in controlled environments free from semantic complexity and contamination, we can project our findings with greater confidence to prevailing benchmarks. However, such validation and projection in mainstream benchmarks remain challenging: data contamination, models possess unknown internalized knowledge, and finding proxies for non-observable variables, such as ground-truth utilization order, is inherently difficult.

Limitations

There are two directions that we did not cover in this paper, which warrant future investigation. First, we did not focus on input that is not a reasoning question, but an open-ended prompt. Second, cer-

tain questions may not contain enough explicit critical information that the reasoning requires, but instead expect the model to infer implicit knowledge. In such cases, does presentation order still matter, or do other factors dominate? We encourage the research community to explore these questions and extend our framework to broader contexts beyond mathematical and programming domains.

Ethics Statement

The datasets that we used in the experiments are publicly available. In our work, we explore factors contributing to problem difficulty for large language models, with a focus on the relationship between information presentation and utilization order, as well as reasoning performance. Our findings aim to improve understanding of LLM capabilities and limitations, which could inform the development of more robust and reliable AI systems. We do not expect any direct ethical concerns from our work.

References

Anthropic. 2025. Claude 3.7 sonnet system card.

Maciej Besta, Julia Barth, Eric Schreiber, Ales Kubicek, Afonso Catarino, Robert Gerstenberger, Piotr Nyczyk, Patrick Iff, Yueling Li, Sam Houliston, and 1 others. 2025. Reasoning language models: A blueprint. *arXiv preprint arXiv:2501.11223*.

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

Yixin Cao, Shibo Hong, Xinze Li, Jiahao Ying, Yubo Ma, Haiyuan Liang, Yantao Liu, Zijun Yao, Xiaozhi Wang, Dan Huang, and 1 others. 2025. Toward generalizable evaluation in the llm era: A survey beyond benchmarks. *arXiv preprint arXiv:2504.18838*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, and 1 others. 2021a. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12039–12050, Bangkok, Thailand. Association for Computational Linguistics.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

OpenAI. 2024. Gpt-4o system card. *https://arxiv.org/pdf/2410.21276*.

Zhenting Qi, Hongyin Luo, Xuliang Huang, Zhuokai Zhao, Yibo Jiang, Xiangjun Fan, Himabindu Lakkaraju, and James R. Glass. 2025. Quantifying generalization complexity for large language models. In *The Thirteenth International Conference on Learning Representations*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Chandan Singh, Jeevana Priya Inala, Michel Galley, Rich Caruana, and Jianfeng Gao. 2024. Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761*.

Peiyang Song, Pengrui Han, and Noah Goodman. 2025. A survey on large language model reasoning failures. In *2nd AI for Math Workshop @ ICML 2025*.

Kv Aditya Srivatsa and Ekaterina Kochmar. 2024. What makes math word problems challenging for LLMs? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1138–1148, Mexico City, Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, and 1 others. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

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

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2793–2804, Mexico City, Mexico. Association for Computational Linguistics.

A Prompt Templates

We provide the full text of the prompt templates in this section.

Extract Critical Information

```
You are given a math problem and its solution.
Extract the key conditions in the problem in the
     order they appear in the solution.
Omit any intermediate variables. Only focus on the
     original variables/conditions. Keep the
     original variable format.
Output them as a Python list of strings only.

Example:

Question: {Example Question}

Solution: {Example Solution}

Extracted: {Example Extracted List}

Now extract information from the following:

Question: "{question}"

Solution: "{solution}"
```

Rephrasing Problem Prompt

```
You are given a math problem and a list of
     conditions.
Rewrite the problem so that the conditions are
     presented in the given order.
Preserve the meaning and the final question.

For example:

Question: {Example Question}
Conditions in order: {Example List of Conditions}
Rewritten: {Example Rewritten Question}

Now rewrite for the following:

Problem: "{problem}"
Conditions in order: {extracted list}

Output the rewritten problem only.
```