LogicGame: Benchmarking Rule-Based Reasoning Abilities of Large Language Models

Jiayi Gui[1†*] , **Yiming Liu**[2*] , **Jiale Cheng**[12†*] , **Xiaotao Gu**[1*] , **Xiao Liu**[12] , **Hongning Wang**[2] , **Yuxiao Dong**[2] , **Jie Tang**[2] , **Minlie Huang**[2‡]

[1]Zhipu.AI [2]Tsinghua University

Abstract

Large Language Models (LLMs) have demonstrated notable capabilities across various tasks, showcasing complex problem-solving abilities. Understanding and executing complex rules, along with multi-step planning, are fundamental to logical reasoning and critical for practical LLM agents and decision-making systems. However, evaluating LLMs as effective rule-based executors and planners remains underexplored. In this paper, we introduce LogicGame, a novel benchmark designed to evaluate the comprehensive rule understanding, execution, and planning capabilities of LLMs. Unlike traditional benchmarks, LogicGame provides diverse games that contain a series of rules with an initial state, requiring models to comprehend and apply predefined regulations to solve problems. We create simulated scenarios in which models execute or plan operations to achieve specific outcomes. These game scenarios are specifically designed to distinguish logical reasoning from mere knowledge by relying exclusively on predefined rules. This separation allows for a pure assessment of rule-based reasoning capabilities. The evaluation considers not only final outcomes but also intermediate steps, providing a comprehensive assessment of model performance. Moreover, these intermediate steps are deterministic and can be automatically verified. LogicGame defines game scenarios with varying difficulty levels, from simple rule applications to complex reasoning chains, in order to offer a precise evaluation of model performance on rule understanding and multi-step execution. Utilizing LogicGame, we test various LLMs and identify notable shortcomings in their rule-based logical reasoning abilities.

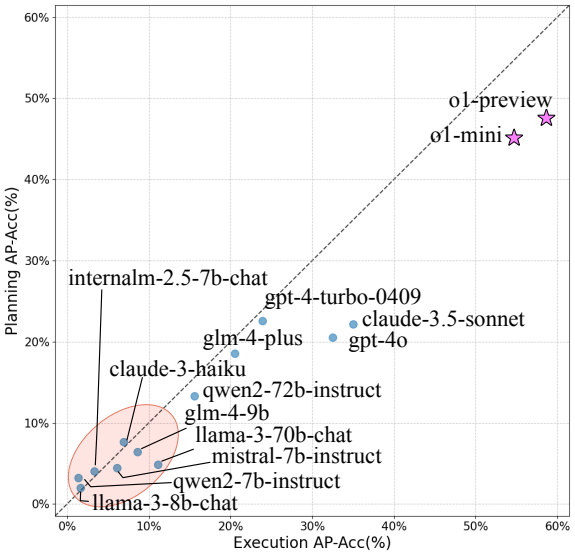

Figure 1: Evaluation results of LogicGame across various models in execution and planning categories. The performance is arithmetic mean of LogicGame's Chinese and English version. Most models struggle on LogicGame getting less than 12% scores in both categories. Two top-performing models highlighted with pink stars stand out.

1 Introduction

Large Language Models (LLMs) have shown notable abilities in a wide range of tasks and even complex problem-solving (Brown et al., 2020; Zeng et al., 2022; Chowdhery et al., 2023; GLM et al., 2024). Their ability to reason, encompassing the understanding of intricate scenarios, strategic planning, and multi-step execution, is crucial for developing advanced AI agents and decision-making systems (Liu et al., 2023; Sumers et al., 2023; Cheng et al., 2024). These capabilities allow LLMs to understand complex user instructions, make logical decisions, and execute tasks accurately.

As alignment becomes integral to the application of LLMs (OpenAI, 2022; Anthropic, 2023; Ouyang

*JG, YL, JC, and XG contributed equally.
†Work done when JG and JC interned at Zhipu AI.
‡Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 1474–1491
July 27 - August 1, 2025

et al., 2022; Cheng et al., 2023) , the primary goal is to align with human intentions and accurately execute their instructions. Simultaneously, these models must possess strong reasoning abilities to handle complicated scenarios. However, evaluating LLMs as effective rule-based executors and planners remains underexplored. Traditional benchmarks usually focus solely on instruction-following or logical reasoning, neglecting the combination of both. Thus, they fail to comprehensively assess the model's reasoning capabilities after elaborate alignment.

In this paper, we introduce **LOGICGAME**, a novel benchmark crafted to evaluate the comprehensive rule understanding, planning, and execution capabilities of LLMs. LOGICGAME offers a set of carefully designed rule-based reasoning games. Each game contains a series of rules the model must follow to find the solution involving single or multiple steps. During the data construction process, we ensure that all the problems remain unavailable on the Internet to prevent data leakage. LOGICGAME covers two main scenarios: execution and planning, each divided into several sub-categories. Execution problems include tasks related to string data manipulation, where models handle string data transformations, as well as arithmetic operations and manipulations, focusing on mathematical computations and sequential execution. Planning games encompass math puzzles, which require solving complex mathematical problems, and pure logic puzzles, involving abstract reasoning without numerical computation. Through these varied games, LOGICGAME aims to comprehensively evaluate the rule-based reasoning capabilities in LLMs. [1]

In LOGICGAME, our goal is to evaluate how well LLMs can reason according to given rules, so we ensure that no additional knowledge is required. The final answer and the process in these scenarios rely solely on the given rules, fostering a pure assessment of the models' rule-based reasoning capabilities. Moreover, the evaluation process in LOGICGAME involves not only the final answers but also the intermediate steps taken by the models, in order to offer a holistic view of the models' performance. Furthermore, process evaluation enables us to determine whether the model faithfully reasons based on established rules rather than merely guessing answers. In addition, these intermediate steps are deterministic and can be automatically verified. To thoroughly assess the rule comprehension and multi-step execution capabilities of various LLMs, LOGICGAME presents problems with multiple difficulty levels. We determine the complexity of each problem by the number of reasoning steps involved. Simple games may require only single-step reasoning, while more challenging game scenarios require multiple reasoning steps, reflecting the need for deeper understanding and more sophisticated reasoning.

By leveraging LOGICGAME, we have conducted extensive experiments across a wide range of LLMs, including api-based models like GPT and GLM families, as well as open-source models such as Qwen and Llama families. Our findings indicate that while LLMs showcase good performance in a variety of tasks, they still exhibit notable shortcomings in rule-based logical reasoning. As illustrated in Figure 1, even the top-performing LLMs achieve around 50% accuracy, with most models scoring less than 12% in both execution and planning categories. Additionally, they struggle significantly with complex reasoning tasks, achieving less than 10% on the most challenging level 3 tasks. Additionally, while few-shot demonstrations can help with execution tasks, they may damage the performance of planning tasks.

Our contributions can be summarized as follows:

- We introduce LOGICGAME, a novel benchmark for rule-based reasoning, including execution and planning tasks, with varying difficulty levels.
- We design an automated assessment process for LOGICGAME, which not only checks the final answers but also analyzes the solution process to comprehensively evaluate LLMs' reasoning abilities.
- We conduct extensive experiments on LOGICGAME across a wide range of LLMs, effectively exposing their deficiencies in rule-based reasoning with the best about 25% overall accuracy.

2 Related Work

The capability to reason has long been a crucial aspect of language models. Research (Wei et al.,

[1]We have released the dev set and the whole set of the LOGICGAME, and created a leaderboard on https://github.com/Hypatiaalegra/LogicGame-Data. You can refer to https://www.codabench.org/competitions/4140/ for a fair and fast evaluation of the LOGICGAME.

Benchmark	Evaluation			Exemplars	Difficulty levels	Data Source
	Process	Verifiable	Determinism			
CLUTRR (Sinha et al., 2019)	✗	✗	✗	✗	✗	semi-synthetic
GSM8K (Cobbe et al., 2021)	✓	✓	✗	✗	✗	human-annotated
PRONTOQA (Saparov and He, 2022)	✓	✓	✗	✗	✗	synthetic
FOLIO (Han et al., 2022)	✗	✗	✗	✓	✓	human-annotated
BIG-Bench Hard (Suzgun et al., 2022)	✗	✗	✗	✓	✗	human-annotated
STRATEGYQA (Geva et al., 2021)	✗	✗	✗	✗	✗	human-annotated
PlanBench (Valmeekam et al., 2024a)	✗	✓	✓	✓	✗	semi-synthetic
Ours	✓	✓	✓	✓	✓	human-annotated

Table 1: Compare LOGICGAME with other logical reasoning benchmarks. BIG-Bench Hard comparison limited to algorithmic part for relevance. Process: Benchmark contains process verification or not. Verifiable: Each step in process verifiable or not. Determinism: Each step in process determined or not. Verifiability and determinism guarantee automated evaluation of process. Exemplars: Examples provided or not.

2022a) has demonstrated that as the size of models increases, their ability to reason emerges, making it a fundamental attribute of LLMs. To elicit this reasoning ability, techniques like chain-of-thought prompting (Wei et al., 2022b) and specialized training (Mukherjee et al., 2023) have become widely adopted. Multi-step reasoning, in particular, is essential for complex decision-making and planning tasks, such as those undertaken by LLM agents (Liu et al., 2023).

Numerous benchmarks have been established over time to rigorously evaluate the reasoning capabilities of neural network models. Early research has concentrated on logical reasoning (Bowman, 2013; Clark et al., 2020; Yu et al., 2020). These studies cover various forms of logic, including inductive, deductive, and abductive reasoning, and aim to assess whether models can infer answers based on given conditions. Mathematical reasoning represents another critical area (Hendrycks et al., 2021; Mishra et al., 2022a). Benchmarks in this domain range in difficulty from grade school problems (Cobbe et al., 2021) to Olympiad-level challenges (Huang et al., 2024) and encompass a variety of formats, from word problems to theory proving (Li et al., 2021; Lample and Charton, 2019). These problems often demand not just reasoning but also robust calculation abilities. Knowledge-based reasoning, particularly commonsense reasoning (Mishra et al., 2022b; Onoe et al., 2021), is another pivotal focus. These benchmarks are designed to determine whether models possess commonsense knowledge and can leverage it to reason effectively. Advancing further, theory-of-mind reasoning (He et al., 2023) examines whether models can understand and incorporate complex layers of human cognition, such as thoughts and beliefs. Plan-based benchmark such as PlanBench(Valmeekam et al., 2024a,b) specifically addresses the planning capabilities of different models. This specificity provides a stringent assessment of a model's planning abilities, although it may not capture a broader range of reasoning processes.

LLMs have undergone extensive alignment, with a key focus on following human instructions. However, reasoning with the capability of instruction-following remains underexplored. Thus, we propose LOGICGAME, a benchmark designed to assess rule-based reasoning, which is a natural integration of logical reasoning with instruction-following capabilities. We compare LOGICGAME with each benchmark in Table 1.

3 LOGICGAME

3.1 Data Construction

3.1.1 Problem Collection and Design

Collection and extraction of rule-based logical tasks. The integration of rule-following and reasoning is a critical aspect of numerous tasks, yet existing benchmarks often fail to adequately capture this complexity. To address this gap, we developed a novel set of problems through extensive research and crowdsourcing. We observed that these tasks share similarities with game mechanics, as they often require adherence to specific rules and decision-making. This insight led us to adopt a gamification approach, which facilitates a nuanced evaluation of models' rule-following and reasoning capabilities.

The top part of Figure 2 shows our categories.

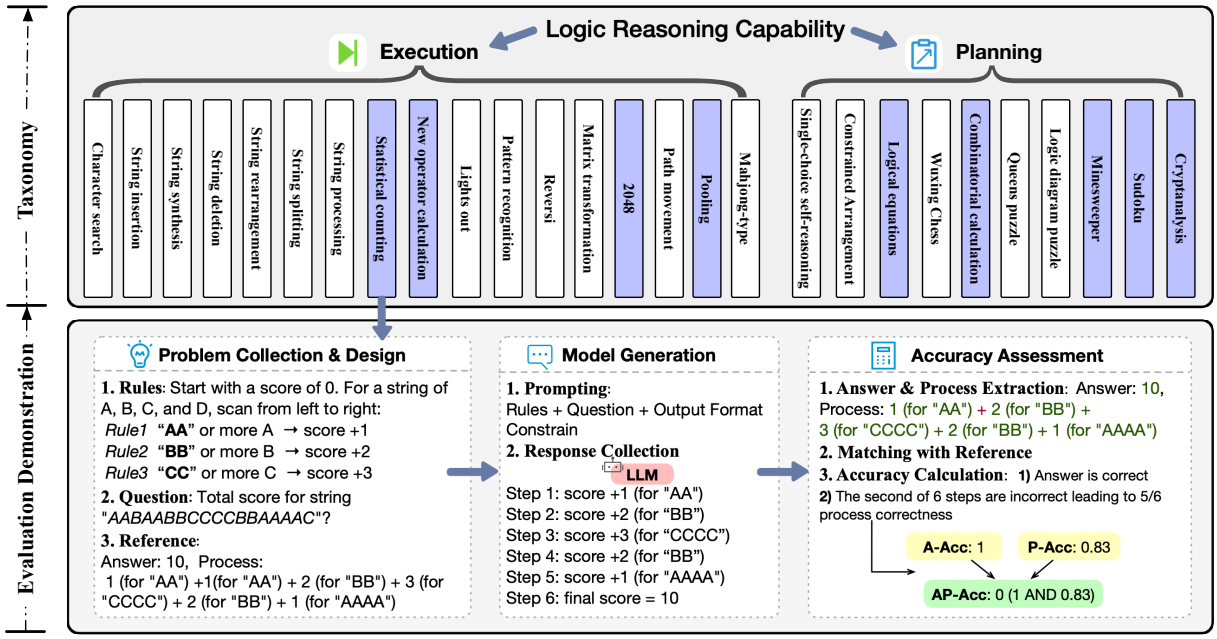

Figure 2: Illustration of taxonomy and evaluation protocol in LogicGame. Taxonomy illustration highlights categories involving mathematics in purple. Json format constrain in evaluation is ommitted due to space limitations and can be referred to Appendix A.

The dataset is structured into two primary domains: Execution and Planning.

Execution domain. In the context of our benchmark, execution refers to a reasoning process characterized by deterministic, single-step inferences. Here, models must apply well-defined rules to manipulate strings or states, with each step yielding a predictable outcome based on the current state and the applied rule. These tasks often require models to execute the correct action from given information, simulating practical scenarios where explicit instructions are given.

Planning domain. The planning domain in our benchmark represents a higher level of complexity, involving long-term strategic thinking and multi-step decision making within rule-governed environments. Unlike traditional "planning" tasks, our planning tasks are atomic and distinct from execution tasks characterized by deterministic, single-step inferences. For example, in a problem like Sudoku, models must consider the interplay of numbers across rows, columns, and grids simultaneously to determine the next move. This move is not predetermined and must be logically inferred. A key feature of tasks in the planning domain is the requirement to use logic and elimination to progress towards a solution. Our focus in this domain is on identifying the correct solution path, rather than optimizing for efficiency, reflecting scenarios where finding any valid solution is the primary goal.

Rule-based problem design and quality control. Following the establishment of our categories, a team of expert human annotators developed problems for each category with a focus on novelty and challenging out-of-domain reasoning, which makes it harder to overfit. To mitigate potential semantic ambiguities associated with natural language reasoning (Fedorenko et al., 2024), we minimized reliance on natural language constructs. Our problems are designed such that the reasoning process does not necessitate natural language inference, allowing for a more direct evaluation of reasoning abilities.

In the execution domain, we ensured that every step is deterministic and verifiable, facilitating precise evaluation and preventing models from resorting to guesswork. For the inherently less deterministic planning problems, we introduced intermediate checkpoints or state variables where appropriate, allowing for a more granular assessment of the problem-solving process. Detailed specifications of these evaluation methods will be provided in Section 3.2.

3.1.2 Output Constraint Design

To facilitate precise evaluation and streamline the matching process, we mandated a structured JSON

Model	Execution					Planning					Overall
	Level 0	Level 1	Level 2	Level 3	Avg.	Level 0	Level 1	Level 2	Level 3	Avg.	
o1-preview	71.11	**57.78**	**48.89**	**51.11**	**57.22**	**70.97**	**54.84**	**41.94**	**22.58**	**47.58**	**53.29**
o1-mini	**73.33**	46.67	44.44	46.67	52.78	**70.97**	48.39	38.71	**22.58**	45.16	49.67
claude-3-5-sonnet	46.67	40.00	33.33	13.33	33.33	64.52	16.13	6.45	6.45	23.39	29.28
gpt-4o	57.78	37.78	31.11	13.33	35.00	48.39	19.35	3.23	3.23	18.55	28.29
gpt-4-turbo-0409	42.22	20.00	17.78	8.89	22.22	51.61	12.90	9.68	3.23	19.36	21.05
glm-4-plus	31.11	15.56	17.78	6.67	17.78	48.39	9.68	9.68	3.23	17.75	17.76
qwen2-72b	28.89	4.44	0.00	0.00	8.33	22.58	6.45	6.45	3.23	9.68	8.88
glm-4-9b	22.22	6.67	2.22	2.22	8.33	22.58	6.45	0.00	0.00	7.26	7.89
internlm2-5-7b	22.22	4.44	4.44	0.00	7.78	12.90	3.23	0.00	0.00	4.03	6.25
claude-3-haiku	8.89	8.89	0.00	0.00	4.45	22.58	0.00	0.00	0.00	5.65	4.93
llama-3-70b	8.89	8.89	8.89	0.00	6.67	3.23	3.23	0.00	0.00	1.62	4.61
mistral-7b	22.22	0.00	0.00	0.00	5.56	9.68	0.00	0.00	0.00	2.42	4.28
qwen2-7b	4.44	2.22	0.00	0.00	1.67	6.45	0.00	0.00	0.00	1.61	1.64
llama-3-8b	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: Performance of 14 models on LOGICGAME of *en* version. The highest performance is **bold**.

output format for model responses. Our evaluation criteria are tailored to the complexity of each problem. For single-step problems categorized as Level 0, models are only required to output the final answer, and evaluation is based solely on the correctness of this answer. However, for problems involving multiple steps or more complex reasoning, which include Levels 1, 2, 3, and certain Level 0 problems, we evaluate both the answer and the process.

In both cases, the output JSON structure includes 'answer', which is a list of strings representing the final solution(s), and for second cases the output also includes 'process', a list of strings detailing each step of the problem-solving process. The details of JSON constraints can be found in Appendix A.

3.1.3 Difficulty Levels and Exemplars

To comprehensively assess models' reasoning capabilities, we have structured our benchmark with four distinct difficulty levels (0, 1, 2, and 3) for each task. The difficulty gradient is determined by two key factors: the complexity of the rules involved and the number of reasoning steps required to arrive at the solution. Each successive level systematically introduces additional rules and reasoning steps. In general, our problems are difficult for models, and some are also challenging for humans.

Furthermore, to evaluate models' capacity for rule acquisition and application, we have developed two distinct exemplars for each question. These exemplars consist of a given question, the correct answer, a step-by-step solution process, and detailed explanations. By providing these examples, we aim to test not only the models' baseline performance but also their ability to learn from demonstrations and apply newly acquired rules to similar problems.

3.1.4 Building Bilingual Benchmark

We developed a comprehensive bilingual benchmark containing both *zh* (Chinese) and *en* (English) versions, with questions in both languages corresponding directly to each other. This approach ensures fairness and broad applicability, allowing our benchmark to more accurately reflect the capabilities of language models across different linguistic backgrounds. The benchmark comprises 304 total items for each language version equally distributed across four difficulty levels (0-3). Each language version containing exactly 76 items per difficulty levels, ensuring balanced representation across all complexity categories.

3.2 Evaluation Protocol

Each model is prompted with a set of rules specific to the given problem, along with a corresponding question and a JSON format constraint for the output, encompassing both the answer and the process, as illustrated in Figure 2. For few-shot trials, example(s) are inserted between the rules and the question to assess the model's in-context learning capabilities. The model responses are then collected and subjected to automated evaluation. As previously mentioned, the evaluation protocol is de-

signed to assess not only the correctness of the answer but also the correctness of the process that led to the answer. Scoring for each problem's answer is determined by comparing the model's response to the reference answer. Similarly, scoring for each problem's process, as defined by the JSON format constraint, is achieved by assessing the degree of alignment between the model's process and the reference process. Specifically, the LOGICGAME defines three metrics related to each problem for scoring:

- **Answer Accuracy(A-Acc)**: This metric evaluates the correctness of the answers for all given questions. It provides an exact match for each answer to indicate whether it is correct. An answer receives a score of 1 if it completely matches the reference answer; otherwise, it scores 0.

- **Process Accuracy(P-Acc):** This metric assesses the correctness of the process, measuring the percentage match based on character-level similarity between the provided process and the expected process. We employ a left-to-right matching approach, given the sequential nature of the reasoning chain that progresses uni-directionally from premise to conclusion. This ensures that each step in the process is evaluated in its correct order. In rare cases where no process is provided in level 0 questions (single-step reasoning), process accuracy is equated to answer accuracy. This approach ensures that all problems have process scores, facilitating fair and consistent calculation across all problems.

- **Answer Process Accuracy(AP-Acc):** This composite metric evaluates the overall accuracy of both the answer and the process. It involves an aggregate score derived by combining **Answer Accuracy** and **Process Accuracy** using a logical AND operation. The AP-Acc receives a score of 1 only when both the answer and the process are exactly correct; otherwise, it scores 0. This reflects that a question is considered correctly solved only when both the reasoning and the resulting answer are accurate.

4 Experiments

4.1 Experimental Setup

We evaluate 14 popular LLMs. Closed-source models included versions from Claude (Anthropic, 2023) and GPT (OpenAI, 2023) series. Open-source models encompassed LLaMA 3 (Touvron et al., 2023), Qwen (Bai et al., 2023), GLM (GLM et al., 2024), Mistral (Jiang et al., 2023), and InternLM (Team, 2023) variants. In the inference stage, we set temperature to 0, ensuring deterministic outputs. The maximum token number is set to 2048. Other parameters are set as their default values.

4.2 Main Results and Analysis

Table 2 presents the performance of 14 LLMs on the English version of LOGICGAME, measured by AP-Acc. The o1-preview model leads with 53.29% overall accuracy for, closely followed by o1-mini. These results underscore the persistent challenge of reasoning for LLMs, as even top performers barely exceed 50% accuracy. The substantial performance gap, ranging from over 50% to below 5%, not only highlights the varying capabilities of current LLMs in complex reasoning tasks, but also emphasizes that despite recent advancements, logical reasoning remains a significant hurdle for most language models. The performance on the Chinese version of the dataset, as shown in Appendix C, demonstrates a relatively similar trend.

The performance drop from Level 0 to Level 3 is not uniform across models. Some models (e.g. o1-mini) show a more gradual decline, while others drop off sharply after Level 1. This may suggest that some models have better consistency across task complexities.

Interestingly, the performance of models in Execution and Planning tasks varies. Some top-performing models, such as o1-preview, demonstrates superior performance in Execution compared to Planning. Conversely, llama-3-70b-chat excels in Planning over Execution. Even within model families, the relative performance differences are evident. In the GPT family, gpt-4-turbo-0409 outperforms gpt-4o in Planning tasks, while gpt-4o shows better performance in Execution tasks.

During the evaluation, it was observed that some models occasionally failed to adhere to the requirement of producing JSON format output. Detailed results and analysis are provided in Appendix B.

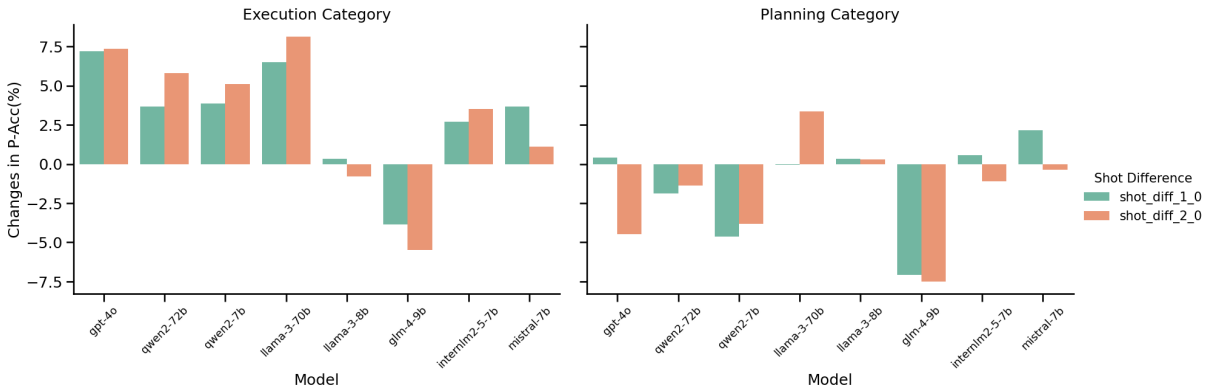

Figure 3: Few-shot differences on execution and planning category of LOGICGAME's *en* version. "shot_diff_1_0" represents the difference in the P-Acc score between the 1-shot and 0-shot settings, calculated as the result of 1-shot minus the result of 0-shot, "shot_diff_2_0" representing the P-Acc score between the 2-shot and 0-shot settings similarly. Positive values represent an improvement in the P-Acc metric when examples(1 or 2) are added to the context, while negative values indicate a decrease. The amplitude of these values signifies the magnitude of improvement or decline.

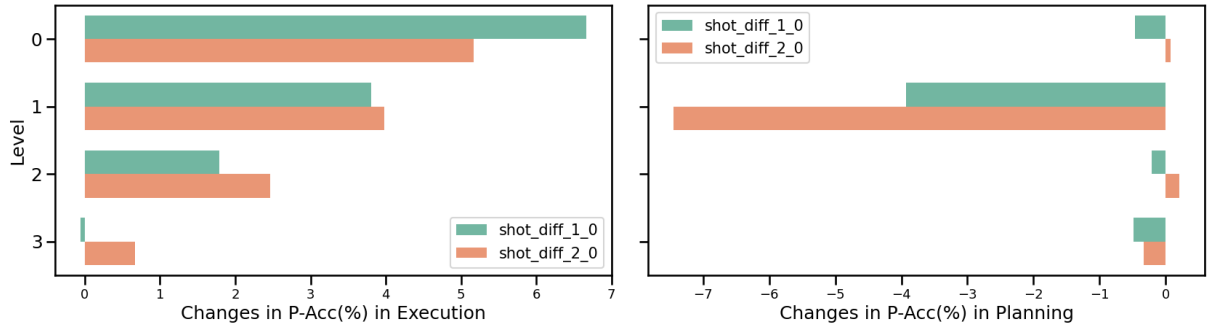

Figure 4: Few-shot differences on difficulty levels of LOGICGAME's *en* version with shot difference settings similar with Figure 3. Positive values denote an improvement in the models' average performance at a certain difficulty levelwhile negative values indicate a decrease.

Despite this, the overall error rates are low across most models, resulting in minimal differences in rankings among those with similar performance.

4.3 Few-shot Results

We conducted experiments to analyze the change of model's performance on 0-shot, 1-shot and 2-shot settings. And models of gpt-4o, qwen2-72b-instruct and qwen2-7b-instruct, llama-3-70b-chat and llama-3-8b-chat, glm-4-9b, mistral-7b-instruct and internlm-2.5-7b-chat are chosen for this trial. The analysis demonstrated in Figure 3 and Figure 4 reveals mixed results of LOGICGAME's *en* version in the "Planning" and "Execution" categories, difficulty levels across the various shot settings. Appendix E shows the results of LOGICGAME's *zh* version.

In the "Execution" category, models demonstrate notable improvements in accuracy with increased shot contexts demonstrated in Figure 3. Specifically, **stronger models** (as indicated in Table 4 and Table 2) like gpt-4o, llama3-70b-instruct **shows a greater increase in the AP-Acc score** when transitioning from **0-shot to 1-shot and 2-shot** settings than weaker ones, indicating enhanced execution accuracy with more contextual information. However, the effects of 1-shot and 2-shot settings vary across models. Performance variations by difficulty levels, as shown in Figure 4, indicate that **models benefit most from 1-shot and 2-shot contexts at Level 0**. And in general, **the influence of shot contexts diminishes as the difficulty level increases**. This consistency suggests that simpler tasks (Level 0) allow models to leverage additional context more effectively, enhancing their execution capabilities across the board.

Conversely, the "Planning" category presents more heterogeneous results. **Models often show declines in performance when moving from 0-shot to 1-shot or 2-shot** settings demonstrated in

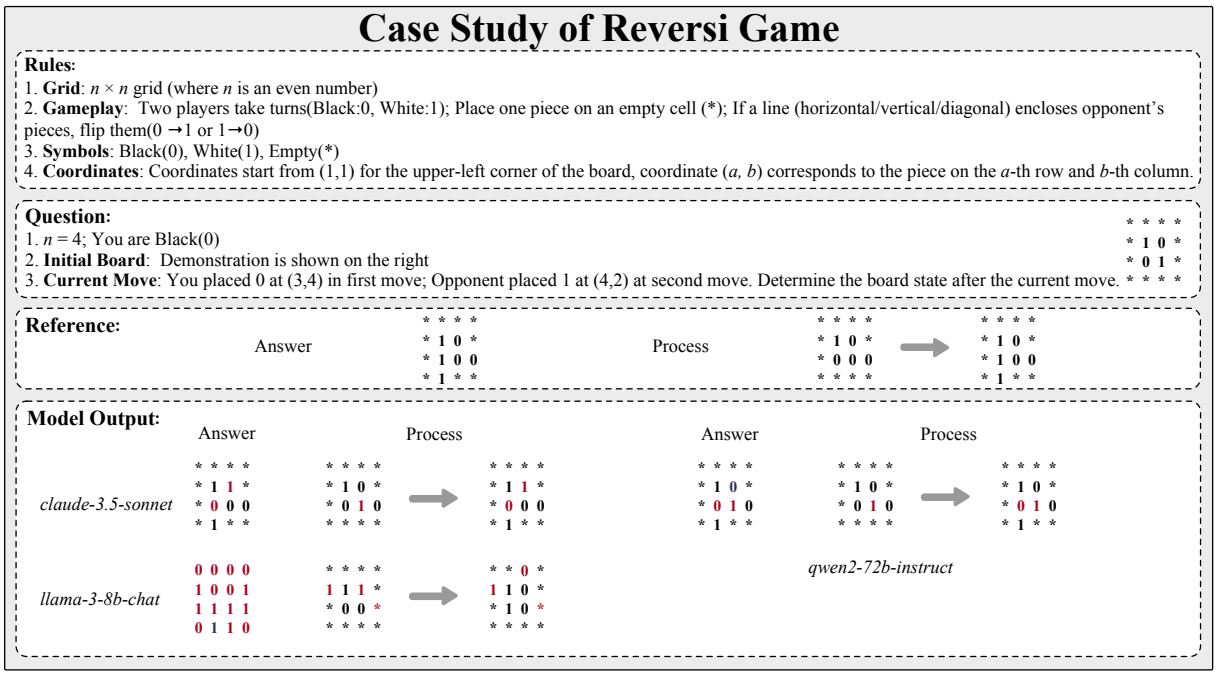

Figure 5: An example of a Reversi game with model outputs, including the answer and process, is shown. The board states for initial, reference, and model outputs are visualized with errors highlighted in red. JSON constraints are omitted due to space, referenced in Figure 6.

Figure 3. These results suggest that while additional context can enhance performance for some models, it can introduce noise for others, potentially obfuscating key elements necessary for planning tasks. Furthermore, Figure 4 illustrates that the **negative impact of both 1-shot and 2-shot contexts are most pronounced at Level 1**. This phenomenon might be because Level 2 and 3 tasks are inherently more challenging, making the performance variability less noticeable, whereas Level 0 tasks are simpler, allowing models to better comprehend the shots, thus resulting in minimal impact. Level 1, however, is most susceptible to disturbance due to its moderate difficulty.

5 Discussion

Case study on Revesi game. From Appendix F, it is evident that all models perform poorly in the Reversi game. Consequently, we conducted a case study on this particular game scenario. The responses from various models tasked with determining the outcome of a Reversi game are analyzed as shown in Figure 5. Despite all models except llama-3-8b-chat adhering to the instruction format and correctly interpreting the initial setup, all models failed to provide the correct answer, demonstrating various types of inaccuracies. The key reasons for failure include:

- **Mismanagement of Details**: For instance, claude-3.5-sonnet misplaced markers or incorrectly transformed pieces, showing that while the general rules were understood, the model failed to apply specific game rules correctly.
- **Inadequate Execution/Planning Understanding**: Models like qwen2-72b-instruct produced incorrect board states following what should have been straightforward captures, revealing a fundamental misunderstanding of the game's piece-flipping mechanisms as well as the initial conditions outlined in the problem.
- **Excessive Alterations**: The llama-3-8b-chat model drastically altered the board state in an unrealistic manner, adding rows and altering more positions than the rules allow, suggesting a misinterpretation of the core principles of the game, particularly with regards to matrix operations and the understanding and execution of piece-flipping mechanisms.

6 Conclusion

In this paper, we introduce LogicGame, a novel benchmark designed to evaluate the rule-based reasoning capabilities of LLMs. LogicGame encompasses multiple difficulty levels, focusing on as-

sessing models' understanding of rules, execution based on these rules, and planning abilities. Moreover, we have developed methods to evaluate both outcomes and reasoning processes, ensuring that models follow the given rules faithfully rather than merely guessing answers. Extensive experiments indicate that current large models still exhibit significant deficiencies in rule-based reasoning tasks. More effort needs to be devoted to further enhancing models' abilities to handle complex reasoning scenarios.

Limitations

Our evaluation is partially influenced by the models' ability to follow the provided instructions. While most models, as shown in Appendix D, performed effectively, a few models, like Llama-3-8b-chat and Mistral-7b-instruct, demonstrated less consistent instruction-following behavior. While these discrepancies did not substantially affect the overall accuracy metrics, they introduce some variability that could influence the evaluation accuracy. Future work could explore the development of automated evaluation methods that reduce dependence on instruction-following capabilities, ensuring a more precise assessment across different models. Additionally, our study evaluated a total of 14 representative models; however, due to resource constraints, some popular models such as Llama-3.1-405B could not be included.

Ethics Considerations

Our benchmark, LOGICGAME, is designed exclusively for research purposes. The AI assistants employed in this study were used solely for language polishing of the paper. When collecting data, annotators were informed of the purpose of the data collection and paid according to the regional standards in place. Since our data primarily focuses on reasoning tasks based on rules, there is no security risks.

References

Anthropic. 2023. Introducing claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Samuel R. Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training. *Annual Meeting of the Association for Computational Linguistics*.

Jiale Cheng, Yida Lu, Xiaotao Gu, Pei Ke, Xiao Liu, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2024. Autodetect: Towards a unified framework for automated weakness detection in large language models. *Conference on Empirical Methods in Natural Language Processing*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *International Joint Conference on Artificial Intelligence*.

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

Evelina Fedorenko, Steven T Piantadosi, and Edward AF Gibson. 2024. Language is primarily a tool for communication rather than thought. *Nature*, 630(8017):575–586.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, D. Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language

models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, E. Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq R. Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir R. Radev. 2022. Folio: Natural language reasoning with first-order logic. *Conference on Empirical Methods in Natural Language Processing*.

Yinghui He, Yufan Wu, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. 2023. Hi-tom: A benchmark for evaluating higher-order theory of mind reasoning in large language models. *Conference on Empirical Methods in Natural Language Processing*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, et al. 2024. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *arXiv preprint arXiv:2406.12753*.

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

Guillaume Lample and François Charton. 2019. Deep learning for symbolic mathematics. *International Conference on Learning Representations*.

Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence Charles Paulson. 2021. Isarstep: a benchmark for high-level mathematical reasoning. *International Conference on Learning Representations*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Yuxian Gu, Hangliang Ding, Kai Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Shengqi Shen, Tianjun Zhang, Sheng Shen, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. Agentbench: Evaluating llms as agents. *International Conference on Learning Representations*.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, S. Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and A. Kalyan. 2022a. Lila: A unified benchmark for mathematical reasoning. *Conference on Empirical Methods in Natural Language Processing*.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Singh Sachdeva, Peter Clark, Chitta Baral, and A. Kalyan. 2022b. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. *Annual Meeting of the Association for Computational Linguistics*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Yasumasa Onoe, Michael J.Q. Zhang, Eunsol Choi, and Greg Durrett. 2021. Creak: A dataset for commonsense reasoning over entity knowledge. *NeurIPS Datasets and Benchmarks*.

OpenAI. 2022. Introducing chatgpt.

R OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *International Conference on Learning Representations*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. *Conference on Empirical Methods in Natural Language Processing*.

T. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive architectures for language agents. *Trans. Mach. Learn. Res.*

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Annual Meeting of the Association for Computational Linguistics*.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. `https://github.com/InternLM/InternLM-techreport`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2024a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024b. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, E. Chi, Tatsunori Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. 2022a. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *International Conference on Learning Representations*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, W. Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, P. Zhang, Yuxiao Dong, and Jie Tang. 2022. Glm-130b: An open bilingual pre-trained model. *International Conference on Learning Representations*.

A JSON prompt

The evaluation JSON constrain prompt template we used to evaluate the performance of different models is shown in Figure 6.

B Data overview of different categories

Table 3 provides a detailed classification of categories along with the corresponding sample sizes as shown below. It should be noted that the quantities presented in Table 3 pertain to only one language version; hence, the actual total quantity is doubled, reflecting the corresponding one-to-one relationship between the *en* and *zh* versions. We have enumerated the descriptions of all categories as follows:

- **Character Search**: Tasks involving decrypting or restoring original strings from encrypted versions, typically with simple transformations.
- **String Insertion**: Tasks involving operation T on strings, where specific patterns trigger character insertions following defined rules.
- **String Synthesis**: Problems involving creating blocks or combinations from given sets of characters (like [A], [B], [C]) under specific rules.
- **String Deletion and Modification**: Tasks involving transforming strings through character deletion or modification based on pattern recognition.
- **String Rearrangement**: Problems involving reversing or rearranging numbers, including special formats like decimals, fractions, and complex numbers.
- **String Processing**: Complex string manipulation involving multiple sequential instructions, often with numbered operations and specific rule sets.
- **Statistical Counting**: Tasks involving pattern recognition and counting in strings of repeated characters, often calculating final scores.
- **New Operator Calculation**: Mathematical calculations involving custom operators with specific computational rules.
- **Element Operation**: Matrix operations based on comparing and manipulating adjacent elements, often involving pattern-based rules for element modification.
- **Pattern Recognition**: Tasks involving finding squares of specific sizes in character matrices, typically searching for unique patterns with minimum side length requirements.
- **Matrix Transformation**: Tasks involving matrix operations on letter-based grids, focusing on pattern recognition and transformation.
- **Path Movement**: Problems tracking position changes based on directional instructions (like i,j,k,l) in a grid system.
- **Single-choice Self-reasoning**: Multi-question problems where answers depend on other questions' answers, requiring logical deduction.

JSON Prompt Template

Please generate a JSON object that follows standard JSON formatting and indentation, containing a field named 'answer'. The 'answer' field should be a list of strings, where each string represents ... The 'process' field should be a list of strings, where each string ...

Example: String Synthesis

Input: Now there are four different types of blocks: [A], [B], [C], and A, which satisfy the following rules:

1. One [A], one [B], and one [C] can be synthesized into one {A}
2. One [A] and one [B] can be synthesized into one [C]

Rule 1, Rule 2, Rule 1, Rule 2... continue cycling through these rules to synthesize until no further synthesis is possible using either rule.

Question: If we currently have four [A], four [B], and three [C], what will be the result after synthesis?

Json Constraints: Please generate a JSON object that follows standard JSON formatting and indentation, containing a field named 'answer'. The 'answer' field should be a list of strings, where each element represents the number of different types of blocks, in the order of [A], [B], [C], A. For example, if there is 1 block of type [A], 0 blocks of type [B], 3 blocks of type [C], it should be represented as ["1", "0", "3", "0"]. The 'process' field should be a list of strings, where each string records the instructions for each step from the initial state to the final state. First output the blocks that need to be synthesized, followed by the "->" symbol, then output the synthesized block, without adding any extra explanations. For example: ["[A] [B] [C] -> {A}", "[A] [B] -> [C]"].

Output:

```
{
  "answer": ["0", "3", "6", "1"],
  "process": [
        "[A] [B] [C] -> {A}",
        "[A] [B] -> [C]",
        "[A] [B] [C] -> {A}",
        "[A] [B] -> [C]"
    ]
}
```

Figure 6: JSON prompt and an example.

- **Constrained Linear Arrangement**: Problems arranging books, students, or class schedules under multiple specific placement constraints.
- **Mutual Generation and Restriction**: Game-like scenarios involving win/lose/draw outcomes with implicit relationship rules.
- **Logical Equations**: Problems assigning numbers to letters under mathematical constraints and relationships.
- **Combinatorial Calculation**: Problems involving numerical inputs requiring analysis of possible combinations.
- **Eight Queens Puzzle**: Problems based on the classic Eight Queens puzzle, involving placing queens on a chess board (marked with 0, 1, or X) without threatening each other.
- **Logic Puzzle**: Matrix-based puzzles requiring number selection following specific logical rules.
- **Minesweeper**: Grid-based problems using numerical hints to locate mines, similar to classic Minesweeper.
- **Standard Sudoku**: Classic Sudoku puzzles requiring number placement in grids while

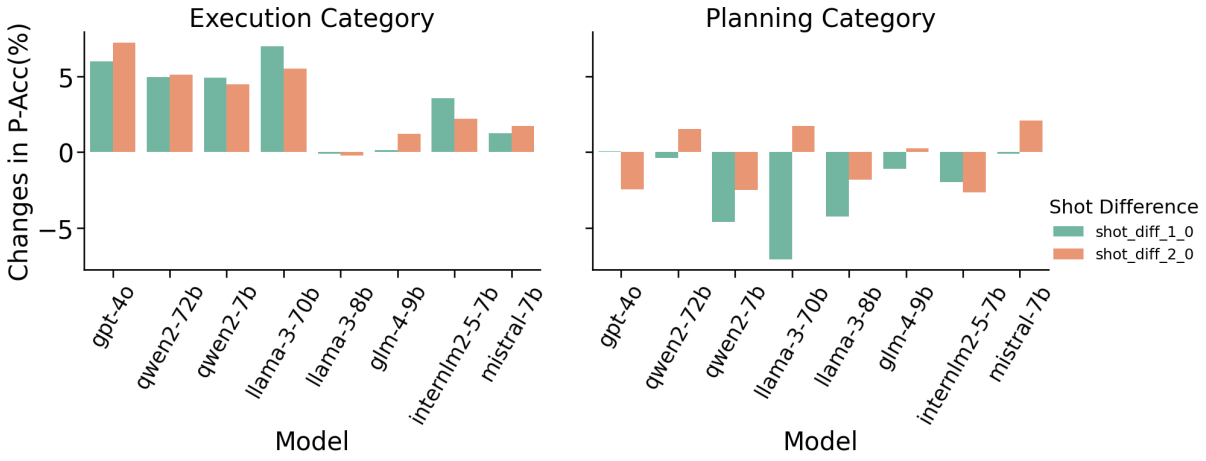

Figure 7: Few-shot differences on execution and planning category of LogicGame's *zh* version. "shot_diff_1_0" represents the difference in the P-Acc score between the 1-shot and 0-shot settings, calculated as the result of 1-shot minus the result of 0-shot, "shot_diff_2_0" representing the P-Acc score between the 2-shot and 0-shot settings similarly. Positive values represent an improvement in the P-Acc metric when examples(1 or 2) are added to the context, while negative values indicate a decrease in the P-Acc metric under the same conditions. The amplitude of these values signifies the magnitude of improvement or decline.

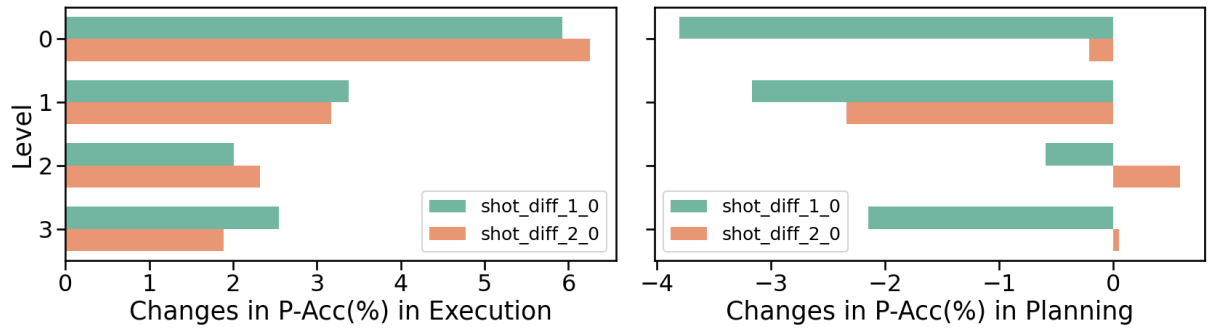

Figure 8: Few-shot differences on difficulty levels of LogicGame's *zh* version with shot difference settings similar with Figure 7. Positive values denote an improvement in the models' average performance at a certain difficulty level in the P-Acc metric when examples(1 or 2) are added to the context, whereas negative values indicate a decline in performance. The amplitude reflects the degree of improvement or deterioration.

following standard Sudoku rules, without additional arithmetic constraints.

- **Cryptanalysis**: Problems involving deducing combinations or values through multiple guesses and feedback.

C Main results of *zh* version

Table 4 presents the performance of 14 LLMs on Chinese version of LogicGame, showcasing both their execution and planning capabilities across different levels of complexity. When comparing the results of the Chinese version to the English version (as shown in Table 5), the o1-preview model consistently performs the best, achieving an overall accuracy of 54.93%, which aligns closely with its leading performance on the English dataset with 53.29% accuracy.

The trend observed in the Chinese version is consistent with the English version, where the performance of models diminishes significantly as the task complexity increases from Level 0 to Level 3. For instance, o1-preview scores 82.22% at Level 0 but drops to 42.22% at Level 3 in terms of execution. Similarly, on the planning aspect, it plunges from 64.52% at Level 0 to 32.26% at Level 3. This pattern highlights the ongoing challenge LLMs face in sustaining high performance across progressively difficult tasks. Overall, while models like o1-preview and o1-mini show competitive performance, most LLMs exhibit a steep decline in accuracy as the task complexity intensifies. The performance disparity between various models remains

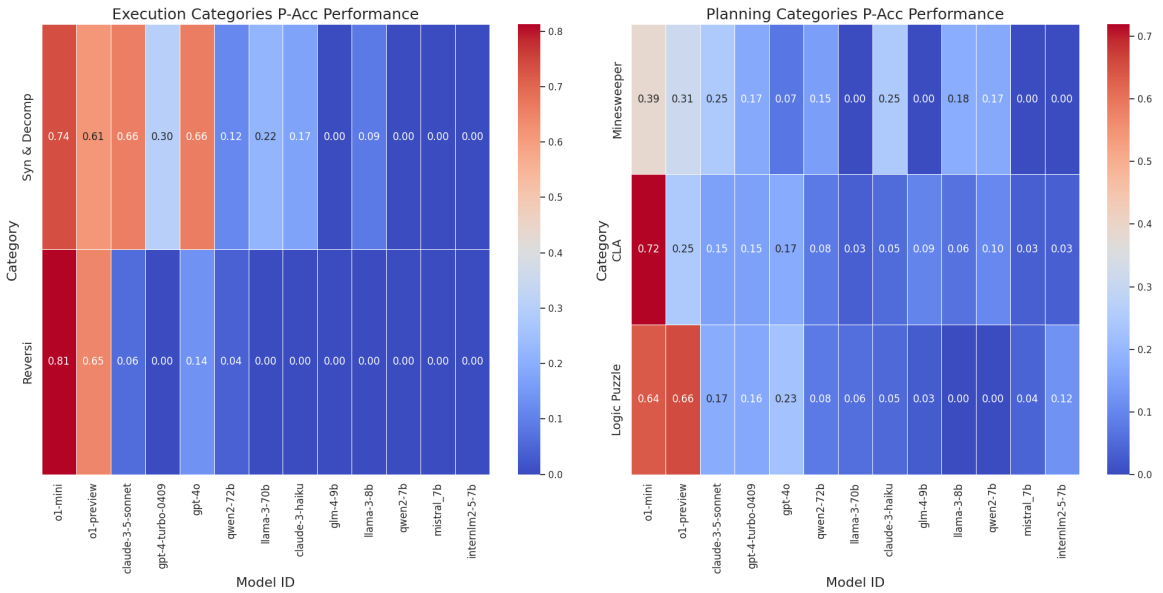

Figure 9: Average AP-Acc scores for five categories with the poorest performance: Reversi and Synthesis and Decomposition from the 'Execution' category, and Minesweeper, Constrained Linear Arrangement, and Logic Puzzle from the 'Planning' category.

substantial, ranging from above 50% to below 5%, thereby underscoring the differing levels of reasoning abilities among current LLMs. Despite certain models demonstrating notable capabilities, logical reasoning continues to be a formidable challenge across both English and Chinese versions.

D Additional evaluation metrics and error analysis

During the evaluation process, it is observed that some models occasionally fail to adhere to instructions regarding the constraint of JSON format output. In this context, we have defined two error metrics and one correctness metric for thorough analysis:

- **JSON Error (JSError):** This metric tracks instances where there is an error in parsing the JSON format, usually due to incomplete or improperly formatted JSON outputs.
- **Instruction Following Error (IFError):** This metric measures the number of instances in which a JSON format could not be successfully extracted.
- **Answer Process Accuracy based on non-IFError and non-JSError (NIJ-Acc):** This is the correctness metric, which evaluates the Answer Process Accuracy (AP-Acc) concerning non-IFError and non-JSError. The formula is provided in Equation 1.

$$S_{\text{NIJ-Acc}} = \frac{S_{\text{AP-Acc}}}{1 - S_{\text{IFError}} - S_{\text{JSError}}} \quad (1)$$

The metrics presented in Table 5 and Table 6 provide a comprehensive assessment of the model's error frequency in *en* and *zh* version respectively. The NIJ-Acc metric evaluates the model's overall accuracy. In this study, the llama-3-8b-chat and llama-3-70b-chat model shows consistent poor instruction-following capabilities in both versions, while the mistral-7b-instruct model performs poorly in the metric of json format. Conversely, stronger models tend to perform more stably as observed in these metrics. Overall error probabilities (IFError, JSError) are low across most models, resulting in minimal variations in NIJ-Acc scores compared to the average scores in Table 2 and Table 4, with only minor rank adjustments among similarly performing models.

E Few shot results of *zh* version

Figure 7 and Figure 8 shows few-shot results of LogicGame's *zh* version. For the "Execution" category, the observed trend aligns with the *en* version, where models exhibit significant improvements in accuracy as the number of shot contexts increases.

However, the impact of 1-shot and 2-shot settings varies among different models. At Level 0 demonstrated in Figure 8, models generally show the most substantial benefit from 1-shot and 2-shot contexts.

For "Planning" category, seen in Figure7, the results are more varied but still consistent with the *en* version. Notably, Figure 8 indicates that the negative impact of 1-shot contexts is most pronounced at Level 0. As the difficulty level rises, the influence of 1-shot contexts diminishes, resulting in smaller performance variations. Conversely, 2-shot contexts show more instability, and no clear pattern emerges from their impact.

F Problems all models fail

Figure 9 categorizes the five areas where all models exhibit the poorest performance, presenting the average AP-ACC scores for each category in a heat map. The horizontal axis corresponds to the model capabilities as outlined in Table 2 and Table 4. This figure highlights that models generally struggle most in two sub-categories within the execution scenario, particularly with 'Reversi', where many models score close to zero except for o1-mini and o1-preview models. Conversely, in planning scenarios such as 'Constrained Linear Arrangement', there is a slight variation in performance across different models.

Categories	Basic Tasks	Application	#Samples
Execution	Character search		12
	String insertion		8
	String synthesis	Synthesis and decomposition	24
	String deletion and modification		8
	String rearrangement		8
	String splitting		8
	String processing	Mahjong-type	16
	Statistical counting$^{\lambda}$		8
	New operator calculation$^{\lambda}$		12
	Element operations	Lights out	20
	Pattern recognition	Reversi	16
	Matrix transformation	2048$^{\lambda}$	24
	Path movement	Pooling$^{\lambda}$	16
Planning	Single-choice self-reasoning		8
	Constrained Linear Arrangement		16
	Mutual generation and restriction		16
	Logical equations$^{\lambda}$		8
	Combinatorial calculation$^{\lambda}$		8
	Eight Queens puzzle	Letter logic diagram	16
	Logic puzzle$^{\lambda}$		12
	Minesweeper$^{\lambda}$		8
	Standard Sudoku$^{\lambda}$	Sudoku with arithmetic rules$^{\lambda}$	16
	Cryptanalysis$^{\lambda}$		16
Total			304

Table 3: An overview of LOGICGAME, encompassing both execution and planning for a single version (either *en* or *zh*). Light red rows indicate sequential-based tasks, light blue rows indicate matrix-based tasks. Tasks involving math are denoted with the superscript$^{\lambda}$.

Model	Execution					Planning					Overall
	Level 0	Level 1	Level 2	Level 3	Avg.	Level 0	Level 1	Level 2	Level 3	Avg.	
o1-preview	**82.22**	**66.67**	**48.89**	**42.22**	**60.00**	**64.52**	**58.06**	**35.48**	**32.26**	**47.58**	**54.93**
o1-mini	77.78	57.78	**48.89**	**42.22**	56.67	74.19	**58.06**	**35.48**	12.90	45.16	51.97
claude-3-5-sonnet	68.89	40.00	22.22	15.56	36.67	48.39	25.81	6.45	3.23	20.97	30.26
gpt-4o	62.22	31.11	13.33	13.33	30.00	51.61	22.58	9.68	6.45	22.58	26.97
gpt-4-turbo-0409	60.00	15.56	15.56	11.11	25.56	67.74	22.58	9.68	3.23	25.81	25.66
glm-4-plus	42.22	26.67	17.78	6.67	23.33	48.39	19.35	6.45	3.23	19.36	21.71
qwen2-72b	53.33	20.00	15.56	2.22	22.78	45.16	16.13	3.23	3.23	16.94	20.39
llama-3-70b	42.22	11.11	6.67	2.22	15.56	25.81	6.45	0.00	0.00	8.07	12.50
claude-3-haiku	31.11	4.44	2.22	0.00	9.44	32.26	6.45	0.00	0.00	9.68	9.54
glm-4-9b	24.44	8.89	2.22	0.00	8.89	19.35	3.23	0.00	0.00	5.65	7.57
internlm2-5-7b	13.33	4.44	0.00	0.00	4.44	16.13	3.23	0.00	0.00	4.84	4.61
llama-3-8b	11.11	2.22	0.00	0.00	3.33	9.68	6.45	0.00	0.00	4.03	3.62
mistral-7b	4.44	0.00	0.00	0.00	1.11	19.35	3.23	0.00	0.00	5.65	2.96
qwen2-7b	4.44	0.00	0.00	0.00	1.11	16.13	3.23	0.00	0.00	4.84	2.63

Table 4: Performance of 14 models on *zh* version. The highest performance is **bold**.

Model	IFError(%↓)		JSError(%↓)		NIJ-Acc(%↑)		
	Execution	Planning	Execution	Planning	Execution	Planning	Avg.
o1-preview	0.56	0.00	0.00	0.00	<u>57.54</u>	<u>47.58</u>	<u>53.47</u>
o1-mini	0.56	0.81	0.56	0.00	53.37	45.53	50.17
claude-3.5-sonnet	0.56	0.00	0.00	0.00	33.52	23.39	29.37
gpt-4o	1.11	0.00	0.00	1.61	35.39	18.85	28.67
gpt-4-turbo-0409	1.11	0.00	0.00	†**2.42**	22.47	19.83	21.40
glm-4-plus	5.00	9.68	1.11	0.00	19.64	18.93	19.22
qwen2-72b-instruct	30.00	16.94	0.56	0.81	12.00	11.76	11.89
llama-3-70b-chat	†**52.22**	‡**60.48**	0.56	0.00	14.12	4.08	10.45
glm-4-9b	21.67	10.48	2.22	0.81	10.95	8.18	9.72
internlm-2.5-7b-chat	15.56	9.68	2.22	0.00	9.46	4.46	7.31
mistral-7b-instruct	19.44	14.52	‡**13.33**	†**2.42**	8.26	2.91	5.80
claude-3-haiku	0.00	0.00	1.11	†**2.42**	4.49	5.49	5.02
qwen2-7b-instruct	2.22	0.81	†**11.67**	‡**3.23**	1.94	1.68	1.82
llama-3-8b-chat	‡**75.00**	†**58.87**	0.00	0.81	0.00	0.00	0.00

Table 5: Model performance on JSError, ResNError, IFError and NIJ-Acc metrics of ***en* version**, with the Avg. calculated as the arithmetic mean NIJ-Acc value of both execution and planning. ‡and †shows the worst and second worst performance in error metrics respectively. <u>Underline</u> shows the best performance in NIJ-Acc metric.

Model	IFError(%↓)		JSError(%↓)		NIJ-Acc(%↑)		
	Execution	Planning	Execution	Planning	Execution	Planning	Avg.
o1-preview	0.00	0.00	0.00	0.00	<u>60.00</u>	<u>47.58</u>	<u>54.93</u>
o1-mini	0.56	0.81	0.56	0.00	56.67	45.16	51.97
claude-3.5-sonnet	0.00	0.00	0.00	0.00	36.67	20.97	30.26
gpt-4o	0.56	0.00	0.00	0.00	30.17	22.58	27.06
gpt-4-turbo-0409	0.56	1.61	0.00	1.61	25.70	26.67	26.09
glm-4-plus	10.00	8.06	0.56	0.00	26.09	21.05	24.00
qwen2-72b-instruct	3.33	1.61	0.56	1.61	23.70	17.50	21.16
llama-3-70b-chat	12.78	†**13.71**	0.00	0.81	17.83	9.43	14.45
claude-3-haiku	3.89	0.81	0.00	0.81	9.83	9.84	9.83
glm-4-9b	25.00	†**13.71**	1.11	0.81	12.03	6.60	9.62
llama-3-8b-chat	‡**40.56**	‡**28.23**	0.00	0.00	5.61	5.62	5.61
internlm-2.5-7b-chat	17.78	1.61	1.67	‡**5.65**	5.52	5.22	5.38
mistral-7b-instruct	†**36.67**	11.29	‡**7.22**	†**4.84**	1.98	6.73	4.39
qwen2-7b-instruct	3.89	2.42	†**2.22**	4.03	1.18	5.17	2.81

Table 6: Model performance on JSError, ResNError, IFError and NIJ-Acc metrics of ***zh* version**, with the Avg. calculated as the arithmetic mean NIJ-Acc value of both execution and planning. ‡and †shows the worst and second worst performance in error metrics respectively. <u>Underline</u> shows the best performance in NIJ-Acc metric.