
LogiPlan: A Structured Benchmark for Logical Planning and Relational Reasoning in LLMs

Yanan Cai*
Microsoft Azure

Ahmed Salem†
Microsoft

Besmira Nushi
Microsoft Research

Mark Russinovich
Microsoft Azure

Abstract

We introduce LOGIPLAN, a novel benchmark designed to evaluate the capabilities of large language models (LLMs) in logical planning and reasoning over complex relational structures. Logical relational reasoning is important for applications that may rely on LLMs to generate and query structured graphs of relations such as network infrastructure, knowledge bases, or business process schema. Our framework allows for dynamic variation of task complexity by controlling the number of objects, relations, and the minimum depth of relational chains, providing a fine-grained assessment of model performance across difficulty levels. LOGIPLAN encompasses three complementary tasks: (1) *Plan Generation*, where models must construct valid directed relational graphs meeting specified structural constraints; (2) *Consistency Detection*, testing models’ ability to identify inconsistencies in relational structures; and (3) *Comparison Question*, evaluating models’ capacity to determine the validity of queried relationships within a given graph. Additionally, we assess models’ self-correction capabilities by prompting them to verify and refine their initial solutions. We evaluate state-of-the-art models including DeepSeek R1, Gemini 2.0 Pro, Gemini 2 Flash Thinking, GPT-4.5, GPT-4o, Llama 3.1 405B, O3-mini, O1, and Claude 3.7 Sonnet across these tasks, revealing significant performance gaps that correlate with model scale and architecture. Our analysis demonstrates that while recent reasoning-enhanced models show promising results on simpler instances, they struggle with more complex configurations requiring deeper logical planning.

1 Introduction

Recent advances in large language models (LLMs) have sparked interest in their ability to perform complex reasoning tasks (Guo et al., 2025; OpenAI, 2025; Jaech et al., 2024). While current work showcases increased reasoning skills in math and coding skills, there are several real-world problems which require reasoning skills that are more algorithmic in nature and often computationally expensive. We still lack an in-depth understanding of how current models perform and how their performance scales with the size of these problems. Logical relational reasoning is a problem in this category, which requires models to plan or validate relationships between a large number of objects. For example, in operations research and business planning for supply chains (Tsouros et al., 2023; Li et al., 2023), there exist several relational planning problems in which each object is a task or an event, and a relationship is a constraint that specifies whether a task should happen before or after another task. Similar examples also exist in code analysis (Neamtiu et al., 2005), and agentic system (Wu et al., 2023; Fourney et al., 2024) where dependencies are introduced and task planning and orchestration is required. The rapid integration of LLMs chatbots and agentic AI systems has transformed how people interact with and manage knowledge through natural language. This evolution necessitates more rigorous evaluation of relational reasoning to better understand the advantages and limitations of LLMs.

In this work, we introduce LOGIPLAN, the first benchmark specifically designed to assess both the generation of structured relational plans and the subsequent reasoning about these plans. Our framework fills an important gap by focusing on the entire pipeline—from the autonomous construction of relational graphs to the critical examination of their logical consistency—which is central to many real-world applications.

*yanacai@microsoft.com

†ahmsalem@microsoft.com

The structured form of LOGIPLAN enables dynamically varying parameters like the number of objects, comparative relations, and the minimum depth of relational chains. This facilitates a fine-grained analysis of model performance across a wide spectrum of difficulties. This approach mirrors the challenges faced by intelligent agents operating in unpredictable environments, where the ability to generate and validate continuously updated plans is critical for successful deployment. Dynamic data generation also offers the advantage that it allows for generating fresh or private copies of the benchmark with controlled difficulty, in cases where benchmark memorization becomes a concern.

LOGIPLAN consists of three complementary tasks: (1) *Plan Generation*, where models must construct valid directed graphs with comparative relationships meeting specified structural constraints. (2) *Consistency Detection*—Does a given graph have inconsistencies or cycles? What are they?; and (3) *Comparison Question*—Is a relationship valid for a given graph?. Through a set of LOGIPLAN experiments, we present a comprehensive evaluation of nine state-of-the-art models, encompassing both instruction-tuned and reasoning-capable LLMs. Our analysis uncovers significant performance disparities that correlate with model scale and architecture, providing valuable insights into the evolving capabilities of LLMs.

Our evaluation reveals a significant gap between the reasoning-based and instruction-based models in the Plan Generation task. This difference stems from the reasoning models’ ability to identify a straightforward generation technique where objects maintain logical relationships, such as $A > B > \dots > Z$. For the Consistency Detection and Comparison Question tasks, the gap persists, primarily between O1 and O3-mini and the other models. In particular, Consistency Detection remains the most difficult task, where even state-of-the-art reasoning models face challenges as the size of the problem increases. Specifically, the performance drop for this task happens much earlier for models like DeepSeek R1 and Gemini 2 Flash Thinking.

Overall, LOGIPLAN serves as both a diagnostic tool and a catalyst for future research, pushing the boundaries of what is achievable in logical relational reasoning with LLMs. We will open source the benchmark, the code for data generation, and the evaluation to enable future research on LLM evaluation and reasoning.

2 LOGIPLAN Construction and Data Generation

Our approach focuses on developing a dynamic benchmark to rigorously probe the logical planning and relational reasoning capabilities of LLMs. The methodology underlying LOGIPLAN is designed to ensure comprehensive coverage of the targeted reasoning tasks while maintaining control over the complexity of the generated relational structures.

2.1 Data Synthesis

We adopt a data generation strategy that creates synthetic relational graphs with controllable complexity. By adjusting parameters, such as the number of objects, relations, and the minimum depth of relational chains, we synthesize tasks that systematically probe the models’ capabilities. This approach allows us to generate diverse and scalable problem instances, ensuring that the benchmark reflects both simple and highly complex reasoning scenarios.

2.1.1 Plan Generation

The Plan Generation task is motivated by real-world applications where models must autonomously construct complex relational structures—such as network topologies or business process models—without human intervention. The primary goal is to evaluate the models’ ability to produce consistent, accurate, and complete plans that adhere to specified constraints.

Task Design. In this task, each model generates a relational graph based on given specifications (i.e., number of objects and relations). The evaluation encompasses several dimensions: the overall accuracy in generating the required relations, the consistency of the constructed

graph in terms of logical coherence, the absence of redundant relations. This rigorous evaluation framework ensures that the models are not only capable of constructing a plan but also of maintaining structural integrity and coherence as the complexity of the task increases.

Data Generation. We generated datasets of varying complexity using a Python program that systematically pairs the number of object with the number of relations. The dataset comprises pairs with object counts ranging from 3 to 50. For each object configuration, we varied the number of relations from minimal (equal to the object count n) to dense (approaching $n(n - 1)/2$, with practical upper limits), creating a diverse testbed for evaluating language model performance on graph-structured generation tasks.

2.1.2 Consistency Detection

In many applications, especially those involving critical infrastructure or knowledge management, identifying inconsistencies such as cycles or contradictory relations is important. The Consistency Detection task is designed to assess a model’s ability to scrutinize a given graph and pinpoint logical errors. Note that the algorithmic complexity of cycle detection in a directed graph is $O((V + E)(C + 1))$ as shown by Johnson (1975), where V is the number of vertices, E is the number of edges, C is the total number of simple cycles in the graph.

Task Design. For this task, models are presented with relational graphs that may contain cycles or contradictions. The evaluation focuses on two main aspects: the model’s ability to detect the presence of inconsistencies and its ability to correctly identify and extract these inconsistencies. This involves assessing the effectiveness in detecting true inconsistencies while minimizing false alarms, and providing a balanced measure of its detection capabilities. By varying the number of objects, relations, and the minimum cycle length (how many steps are needed for the cycle to close), we challenge the models with increasingly complex scenarios, ensuring a comprehensive assessment of their verification capabilities.

Data Generation. We used the Python NetworkX library to construct controlled relational graphs, encompassing both positive cases (acyclic, consistent graphs) and negative cases (cyclic, inconsistent graphs). The algorithms generate directed graphs with parameterized complexity, varying the number of objects and the density of relations while ensuring complete connectivity. For cyclic cases, we enforce minimum cycle lengths in the graph to assess multi-step reasoning abilities. These graphs are then transformed into randomized textual statements where: (1) objects receive arbitrary labels unrelated to their structural positions, (2) relations are inconsistently expressed using both forward and backward notation (randomly choosing between $>$ or $<$), and (3) statements are deliberately presented in shuffled order. This intentional randomization prevents models from solving tasks through superficial pattern recognition, instead requiring logical reasoning to reconstruct the complete relation network and identify cycles regardless of their surface representation. As an example, Figure 6 in Appendix illustrates an example of the graph used in the benchmark dataset.

2.1.3 Comparison Question

The Comparison Question task simulates scenarios where models must interpret a pre-existing relational graph and make informed judgments about specific relations. This task is pivotal for applications such as automated reasoning in databases or dynamic knowledge base querying. Note that the algorithmic complexity of the reachability check between two nodes in a directed graph is $O(V + E)$ using Breadth-First Search (BFS) or Depth-First Search (DFS).

Task Design. In this task, models are required to evaluate relational statements and categorize them as “True”, “False”, or “Unknown”. The Unknown category is particularly significant as it assesses a model’s ability to handle incomplete information, i.e., situations where the given graph is insufficient to answer the query about the specified relation. We challenge the models with increasingly complex problems by examining performance based on the number of objects, relations, and the required inference depth. This design tests not

only direct reasoning capabilities but also the model’s decision-making process in complex and edge cases.

Data Generation. We use the same graph generation algorithm as the Consistency Detection task. For each graph, we randomly select object pairs at specific path distances and randomly determine whether to present them as true or false comparisons. A comparison is marked as “True” when it accurately reflects the graph relationship, and “False” when it contradicts it. “Unknown” pairs are specifically created from objects with no connecting paths.

2.1.4 Self Correction

In our benchmark, we also evaluate the models’ self-correction capabilities. A model’s ability to self-assess and correct its output when necessary can demonstrate a logical trait, ensuring that it does not merely alter its responses when questioned by the user.

Task Design. Following the completion of the Consistency Detection and Comparison Question tasks, models are given a follow-up prompt: *Are you sure?*. We assess two key aspects: the degree to which initial decisions are revised and the impact of these revisions on overall accuracy. Specifically, we examine whether self-correction leads to improved outcomes or introduces additional errors. This evaluation sheds light on how internal confidence calibration mechanisms influence the decision-making process, highlighting the model’s capability to adapt and refine its outputs in response to uncertainty.

3 Evaluation

In this section, we present a comprehensive evaluation of our LOGIPLAN framework across a diverse array of state-of-the-art models.

3.1 Metrics

To rigorously evaluate the performance of different models on our tasks, we employ a diverse set of metrics tailored to each task’s unique requirements. This allows us to deeply investigate the models’ performance across varying computational demands.

Plan Generation. For the Plan Generation task, we primarily use *Accuracy*, which measures the overall success rate of the model in generating correct plans. A perfect accuracy score (100%) implies that the model correctly generates the required number of relations with the correct number of objects, and without any inconsistencies.

Specifically, we use more granular metrics to determine the overall *Accuracy*:

- *Consistency*: This metric evaluates whether the generated relations are free from logical inconsistencies. A consistency score of 100% indicates no inconsistencies.
- *Duplicates*: This measures the presence of duplicate relations. A score of 100% means no duplicates are detected.
- *Object Numbers* and *Relation Numbers*: These metrics assess whether the generated objects and relations match the specified numbers in the task. A perfect score of 100% indicates full compliance with the task requirements.

Consistency Detection. For the Consistency Detection task, we use *F1 score* calculated from *Precision* and *Recall* to assess the correctness of the model’s decisions. We evaluate two aspects: (1) whether the model correctly identifies the presence or absence of inconsistencies, and (2) if inconsistencies are present, whether the model correctly identifies and outputs all of them. Specifically:

- *Precision*: For consistency cases, *Precision* measures the correctness of the model reporting consistency. For inconsistency cases, it measures the proportion of correctly identified inconsistencies among all inconsistencies reported by the model.

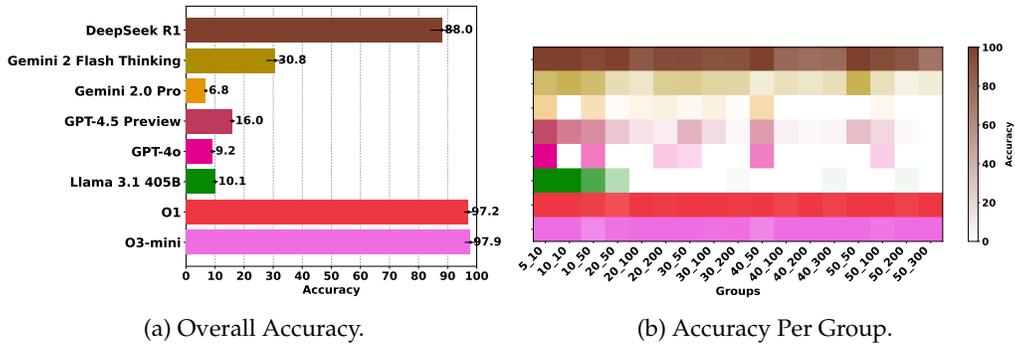


Figure 1: Overview Accuracy of different models for the Plan Generation task.

- *Recall*: For consistency cases, *Recall* is always 1 (as there is no inconsistency to find in the ground truth). For inconsistency cases, it measures the proportion of actual inconsistencies in the ground truth that were correctly identified by the model.

Comparison Question. For the Comparison Question task, we measure *Accuracy* to assess the model’s ability to correctly determine the veracity of given relations. The potential model outputs are as follows:

- “True”: Indicates that the comparison statement can be inferred from the graph.
- “False”: Means the comparison can be inferred from the graph, but the statement is in the opposite direction.
- “Unknown”: Implies that the relation between the two objects in the comparison statement cannot be determined from the graph. This option aims to evaluate the model’s effectiveness in handling ambiguous or incomplete information.

Self Correction. Finally, across the Consistency Detection and Comparison Question tasks, we evaluate the model’s ability to self-correct its output by measuring the change in its decisions when prompted with the query, “Are you sure?”. This metric reveals whether the model’s confidence adjustments lead to more accurate or erroneous outcomes, providing insights into its self-assessment capabilities.

3.2 Evaluation Settings

For all of our experiments (unless explicitly mentioned otherwise), we conduct five independent runs and report the average performance to account for the models’ inherent non-deterministic nature and increase the robustness of our results.

We evaluate the following models under their respective settings. The detailed configuration for each model is presented in the Appendix (Table 1).

Instruction-based models. Llama 3.1 405B, Gemini 2.0 Pro, GPT-4o, GPT-4.5.

Reasoning models. O1, O3-mini, DeepSeek R1, Gemini 2 Flash Thinking, Claude 3.7 Sonnet.

3.3 Results

We now present the evaluation results for LOGIPLAN’s different tasks.

3.3.1 Plan Generation

We begin with the Plan Generation task as a warm up task. In this task, each model generates a consistent list of relations based on the number of objects and relations specified in the prompt, evaluated over three independent runs. Our dataset comprises 142 examples, with objects ranging from 3 to 50 and relations ranging from 3 to 300. For each group of object numbers, the number of relations is incrementally increased from a minimum edge with a

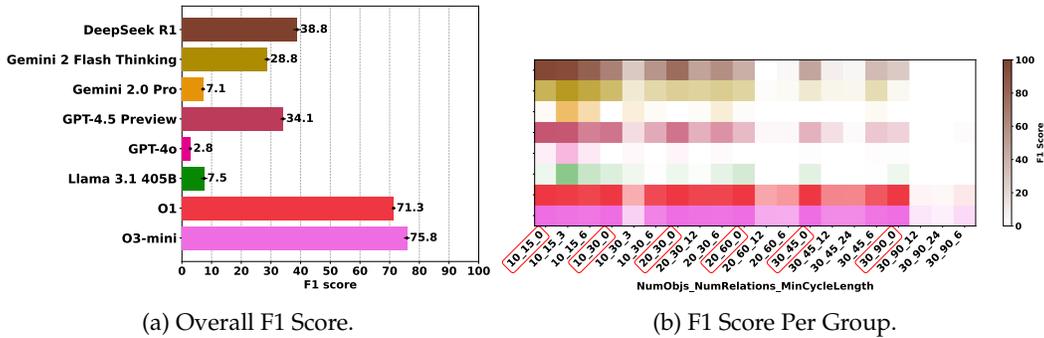


Figure 2: Overview F1 Score of different models for the Consistency Detection task.

minimal step to a maximum reach evenly. Detailed configurations of these 142 examples and prompt template are provided in [Appendix A](#).

We first present the overall accuracy (average across all runs) for each model, as shown in [Figure 1a](#). The figure highlights a substantial performance gap between reasoning models and instruction-based ones. For instance, the O3-mini and O1 models achieve notable accuracies of 97.9% and 97.2%, respectively, while DeepSeek R1 attains 88.0%. Interestingly, the Gemini 2 Flash Thinking model, despite being a reasoning model, only achieves 30.8%, which is significantly lower. However, it still performs approximately twice as well as the best instruction model GPT-4.5. We further check the Gemini 2 Flash Thinking’s failed cases, the main issue is its ability to avoid the presence of duplicates in its output.

Next, we present a more fine-grained analysis of accuracy by dividing the settings into bins and reporting each model’s performance, as illustrated in [Figure 1b](#). In this figure, the x-axis is labeled as x_y represents groups covering the number of objects between $(x_{i-1}, x_i]$ and the number of relations between $(y_{i-1}, y_i]$. This analysis reveals that reasoning models consistently exhibit strong performance, even as the number of relations and objects increases. Conversely, instruction-based models perform well only with limited numbers of relations and objects. In particular, when the number of objects is fixed, instruction-based models consistently perform worse as the number of relations increases.

Manual inspection of the reasoning models’ outputs revealed a noticeable pattern; these models tend to employ a straightforward algorithm for ordering objects, such as $A > B > C > D$, and so on. This strategy allows the models to generate an arbitrary number of consistent relations. To further test their robustness, we provided a list of objects as random strings. Remarkably, the models maintained the simple ordering algorithm and produced consistent relations, demonstrating their robustness in maintaining logical consistency. Detailed plots are presented in [Appendix B](#). Albeit this is only one example, from an algorithm discovery perspective, the fact that models were able to figure out a straightforward solution for the generation, is an interesting indication that longer step-by-step traces can practically facilitate the discovery and execution of algorithmic approaches to problem solving.

3.3.2 Consistency Detection

Next, we evaluate the Consistency Detection task. For this task, we consider 20 different groups, each defined by varying numbers of objects, relations, and the minimum depth of cycles. For example, in a list containing cycles such as $A > B > C > A$, $A > B > A$, and $B > D > A > B$, the minimum cycle depth is 2, with objects A and B involved in the shortest cycle $A > B > A$. For consistent relations, we set the minimum cycle length as 0 in the group name. For each group, we generate 20 different plans, resulting in a total of 400 samples. For each sample, the model is tasked with detecting any cycles/contradictions and, if present, outputting all of them. More details on the data groups and the prompt template can be found in [Appendix A](#).

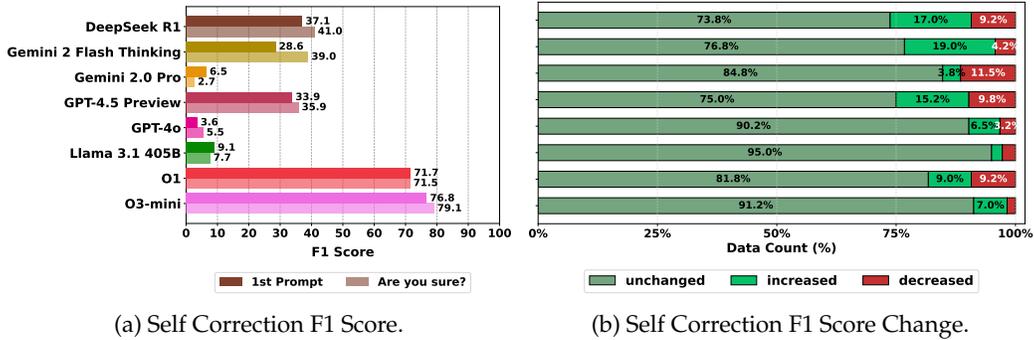


Figure 3: Self correction results of the different models for the Consistency Detection task.

Average F1 scores are shown in Figure 2a. Unlike the Plan Generation task, this task is more challenging for both reasoning and instruction-based models. The performance gap between the O3-mini and O1 models and the next best model (DeepSeek R1) is significant, almost 2x. Interestingly, in this task, GPT 4.5 achieves better performance than Gemini 2 Flash Thinking, indicating that an instruction-based model can outperform a reasoning model. However, the remaining instruction-based models perform significantly worse (by more than 3.8x) than the closest reasoning model.

Next, we report the fine-grained F1 scores for each model and configuration independently in Figure 2b. In this figure, the x-axis is labeled as x_y_z represents groups covering the number of objects, the number of relations, and the minimum cycle length in the graphs. Note that every graph has a randomized numbers of cycles, with the enforcement on the minimum cycle length. The average number of cycles across all samples is 4.37. Similar to the Plan Generation task, the larger the number of objects and relations, the more challenging the task becomes for the models. We also observe that longer cycles increase the difficulty for the models. This is evident when comparing settings with a minimum length of 0 (highlighted with red rectangles in the figure) to the two subsequent columns.

Finally, we report the self-correcting performance of the models for the Consistency Detection task after prompting them with, “Are you sure?”. To reduce the cost, we perform one run with all samples for all models. Figure 3a shows that most models improve their performance upon re-evaluation, with Gemini 2 Flash Thinking achieving a notable gain of over 10%. However, some models exhibit minimal change, such as O1, which shows only a 0.02% reduction, while others experience a performance drop, like Llama 3.1 405B and Gemini 2.0 Pro, by 1.4% and 3.8%, respectively, the latter corresponding to a factor of 2.4x.

Figure 3b provides a detailed breakdown of the changes in F1 scores for each model. The percentage here is calculated across all model outputs, regardless of their correctness. Most models exhibit no significant change in decision most of the time ($\geq 75\%$). Notably, Gemini 2 Flash Thinking shows the highest increase in F1 score at 19% of the samples, while Gemini 2.0 Pro demonstrates the most considerable decrease at 11.5% of the samples.

3.4 Comparison Question

Finally, we evaluate the Comparison Question task. In this task, models are provided with a list of relations and a comparison statement, and are asked to determine whether the statement is “True”, “False”, or “Unknown”. For this task, we only consider consistent graphs. During data generation, we ensure diversity in graph topology, object names, and the number of greater than ($>$) and smaller than ($<$) relations. We categorize the data into 23 different groups, each defined by varying numbers of objects, relations, and the minimum steps required to prove the relation in the question. In the group names, a depth of comparison of 0 indicates that the relation cannot be determined, hence the answer should be “Unknown”. This setup results in a total of 460 samples. More details on the data groups and the prompt template can be found in Appendix A.

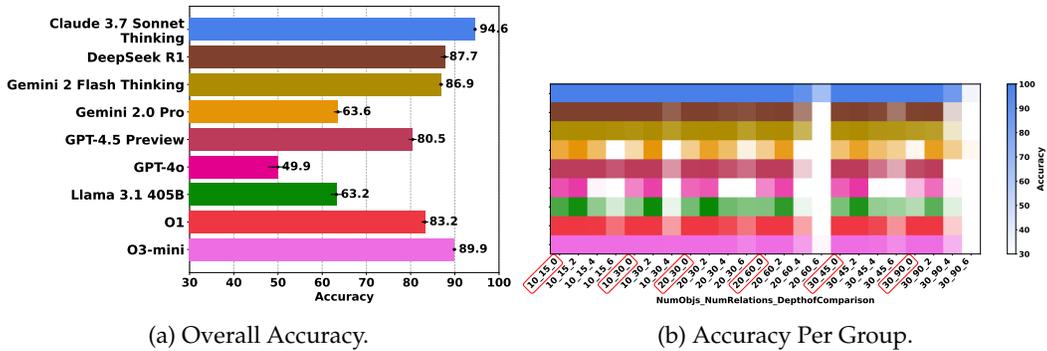


Figure 4: Overview Accuracy of different models for the Comparison Question task.

Figure 4a displays the overall accuracy. Interestingly, for this task, the O1 model drops from its second-place ranking in the previous tasks to below the Gemini 2 Flash Thinking model. Additionally, the top-performing models are much closer in performance for this task, with less than a 10% gap between the first and fifth models. Notably, GPT 4.5 outperforms other instruction-based models, and indicates a smaller performance gap between reasoning and instruction-based models in this task. However, it is important to note that the baseline for random guessing is $\frac{1}{3}$; thus, while GPT-4o achieves 49.9% accuracy, this is only approximately 16.6% better than random guessing.

Next, we present the fine-grained grouped results in Figure 4b. In this figure, the x-axis is labeled as x_y_z represents groups covering the number of objects, the number of relations, and the depth of the comparison inference steps. Specifically we name the groups with “Unknown” comparison statement as 0 at z . Similar to the other tasks, as the number of objects and relations increases, the task becomes more challenging. It is also noteworthy that for instruction-based models such as Llama 3.1 405B, GPT-4o, and Gemini 2.0 Pro, it is easier to determine the correct answer when it is not “Unknown”, i.e., when the depth of comparison is greater than 0 in the chart.

We also plot the models’ self-correction results in Figure 5a. To reduce the cost, we perform one run with all samples for all models. Figure 5a displays both the original results and those obtained after asking, “Are you sure?”. Surprisingly, for this task, Gemini 2.0 Pro, which performed the worst in self-improvement during the Consistency Detection task, shows the best improvement, with a gain of 17.8%, while Llama 3.1 405B continues to perform poorly after the self-check, with a drop of 11.5%. Other models slightly improved, except for DeepSeek R1 and Gemini 2 Flash Thinking, which showed slight degradations.

Figure 5b illustrates the fine-grained changes for all models. Additionally, we highlight instances where the model predicts “Unknown” and the correct label is **not** “Unknown”. As the figure shows, the instruction-based models exhibit the most incorrect changes to “Unknown”, with Llama 3.1 405B and GPT-4o incorrectly converting 14.3% and 6.7% of the correct decisions to “Unknown”, respectively.

4 Related Work

Logical Reasoning in LLMs. Logical reasoning has been studied as a fundamental capability of large language models in the context of inductive reasoning, propositional and first-order logic (Patel et al., 2024; Ryu et al., 2025; Parmar et al., 2024; Xu et al., 2024), and a diverse set of logical puzzles (Lin et al., 2025; Shah et al., 2024). Similarly to LOGIPLAN, these works often involve one or multi-hop reasoning as a way of testing whether models can retrieve and reason upon more than one statement or fact presented in context (Trivedi et al., 2022; Yang et al., 2018; Patel et al., 2024; Schnitzler et al., 2024). For example, HotpotQA includes a set of linguistic facts in each example, and then asks related factoid comparison questions. However, recent work has also shown that the accuracy of models in conducting multi-hop reasoning drops as the length of context increases (Levy et al., 2024; Balachandran et al., 2024) or with the number of inference rules (Patel et al., 2024).

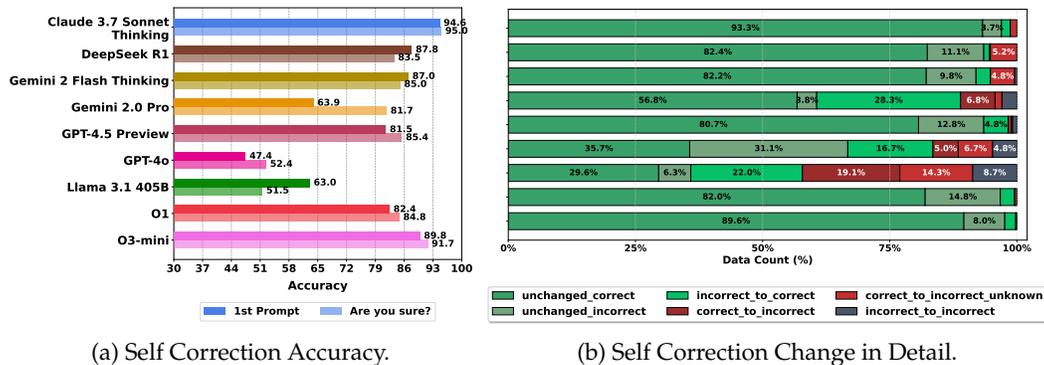


Figure 5: Self correction results of the different models for the Comparison Question task.

Relational reasoning in LLMs has not been studied extensively yet since until recently the task was beyond the capabilities of the state-of-the-art, especially for larger problem sizes. Preliminary work (Li et al., 2024; Alotaibi et al., 2024) with a few-hop queries has shown that larger models can conduct simple logical queries on graphs to extract relationships between objects or their attributes. Advancements in test-time compute and post-training (or self-training) via Reinforcement Learning (RL) (Lightman et al., 2023; Wang et al., 2024c; Gulcehre et al., 2023) have shown to significantly improve general reasoning skills for problems like math and coding (Guo et al., 2025; OpenAI, 2025; Jaech et al., 2024). Qualitatively, these methods lengthen the generations of models in the form of extended reasoning traces, which contain notions of self-reflection, backtracking, and exploration. In this work, we include several of the state-of-the-art reasoning models, to better understand whether these techniques generalize to more complex forms of reasoning, such as inference and cycle detection in larger graphs. More relatedly to our work, Abbe et al. (2024) has shown that intermediate steps in detailed scratchpads in transformers can help address barriers introduced by high connectivity and longer chains in difficult queries.

Planning in LLMs. Our work can also be situated in the context of studying planning abilities of LLMs, which has been evaluated extensively in previous work (Valmeekam et al., 2024; Wang et al., 2024b; Zheng et al., 2024) with benchmarks like PlanBench (Valmeekam et al., 2024), NaturalPlan (Zheng et al., 2024), CogEval (Momennejad et al., 2023). CogEval in particular also looks at routing problems in graphs, but does not extend to cycle detection or consistency checks of relationships. Wang et al. (2024a) instead formulates a task in a multimodal setting that asks multiple-choice questions about the relationships between pairs of objects, which can be inferred from other relationships already presented in context.

Most importantly, given the high computational complexity of the problems in LOGIPLAN, our benchmark also contributes to an emerging line of work that studies how transformers solve NP-hard and NP-complete problems (Fan et al., 2023). With the increased generality of current architectures, it is important to understand in detail the limitations of transformers in solving such complex problems and how they scale with the difficulty of the problem.

5 Conclusion

In conclusion, our work demonstrates that while current LLMs have made notable strides in handling structured relational reasoning, significant performance gaps remain—particularly as task complexity increases. LOGIPLAN not only reveals the limitations of instruction-based models in generating logically consistent plans but also highlights the robust, albeit varied, performance of dedicated reasoning models. This benchmark serves as both a diagnostic tool and a catalyst for future research, guiding the development of more sophisticated architectures and self-correction strategies that can better tackle real-world, complex logical planning problems.

Table 1: List of models studied in this study and corresponding temperature and maximum token limits used for all experiments.

Model	temp.	max token	reasoning
Claude 3.7 Sonnet Thinking 2025-02-19 (Anthropic, 2025)	1.0	32,768	y
DeepSeek R1 (Guo et al., 2025)	0.6	32,768	y
Gemini 2.0 Pro Exp 2025-02-05 (Google, 2025a)	1.0	4,096	n
Gemini 2 Flash Thinking Exp 2025-01-21 (Google, 2025b)	1.0	32,768	y
O1 2024-12-17 (Jaech et al., 2024)	NA	32,000	y
O3-mini 2025-01-31 (high) (OpenAI, 2025)	NA	32,000	y
GPT-4o 2024-08-06 (Hurst et al., 2024)	1.0	4,096	n
GPT-4.5 Preview 2025-02-27 (Hurst et al., 2024)	1.0	4,096	n
Llama 3.1 405B (Dubey et al., 2024)	1.0	4,096	n

Reproducibility Statement

Implementation. All our experimental pipelines are implemented using an open-source evaluation framework. We will make all data generation scripts publicly available upon publication, which will also enable sampling other versions LOGIPLAN with different problem sizes. The open source framework we use supports parameterized scripts that not only facilitate reproducibility but also allow for the generation of more challenging and customized setups tailored to specific scenarios, such as particular object names and varying plan lengths. Additionally, the inference configurations used for all tasks are presented in Table 1, and the prompts used for each task are detailed in Appendix A. We experiment with different rephrasing of the prompts but found no significant effect on the outcomes. For all experiments we run, Top_p is set to 0.95, presence_penalty is set to 0 by default if the model supports this.

Note on experimental results. Our experiments on Gemini 2.0 Pro for Comparison Question were interrupted after four runs as the model was softly deprecated upon the release of Gemini 2.5 Pro on March 25, 2025. Tasks Plan Generation and Consistency Detection instead include results for five runs for Gemini 2.0 Pro. Additionally, due to restrictive rate limiting imposed by Claude, we were unable to complete all planned runs with Claude models before this submission. The result presented in Comparison Question is only based on one run on Claude 3.7 Sonnet with thinking model enabled.

References

- Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the globality barrier and inductive scratchpad. *Advances in Neural Information Processing Systems*, 37:27850–27895, 2024.
- Fatimah Alotaibi, Adithya Kulkarni, and Dawei Zhou. Graph of logic: Enhancing llm reasoning with graphs and symbolic logic. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 5926–5935. IEEE, 2024.
- Anthropic. Claude 3.7 sonnet. <https://www.anthropic.com/news/claude-3-7-sonnet>, 2025. Accessed: 2025-03-17.
- Vidhisha Balachandran, Jingya Chen, Neel Joshi, Besmira Nushi, Hamid Palangi, Eduardo Salinas, Vibhav Vineet, James Woffinden-Luey, and Safoora Yousefi. Eureka: Evaluating and understanding large foundation models. *arXiv preprint arXiv:2409.10566*, 2024.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*, 2023.

-
- Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.
- Google. Gemini 2.0 pro experimental. <https://deepmind.google/technologies/gemini/pro/>, 2025a. Accessed: 2025-03-17.
- Google. Gemini 2.0 flash thinking. <https://deepmind.google/technologies/gemini/flash-thinking/>, 2025b. Accessed: 2025-03-17.
- Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Donald B Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.
- Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *ACL*, 2024.
- Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. Large language models for supply chain optimization. *arXiv preprint arXiv:2307.03875*, 2023.
- Zhiming Li, Yushi Cao, Xiufeng Xu, Junzhe Jiang, Xu Liu, Yon Shin Teo, Shang-Wei Lin, and Yang Liu. Llms for relational reasoning: How far are we? In *Proceedings of the 1st International Workshop on Large Language Models for Code*, pp. 119–126, 2024.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.
- Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.
- Ida Momennejad, Hosein Hasanbeig, Felipe Vieira Frujeri, Hiteshi Sharma, Nebojsa Jojic, Hamid Palangi, Robert Ness, and Jonathan Larson. Evaluating cognitive maps and planning in large language models with cogeval. *Advances in Neural Information Processing Systems*, 36:69736–69751, 2023.
- Iulian Neamtiu, Jeffrey S. Foster, and Michael W. Hicks. Understanding source code evolution using abstract syntax tree matching. *ACM SIGSOFT Software Engineering Notes*, 30:1 – 5, 2005.
- OpenAI. Openai o3-mini system card. <https://openai.com/index/o3-mini-system-card/>, 2025. Accessed: 2025-03-17.
- Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *ACL*, 2024.

-
- Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *arXiv preprint arXiv:2406.17169*, 2024.
- Hyun Ryu, Gyeongman Kim, Hyemin S Lee, and Eunho Yang. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. *ICLR*, 2025.
- Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. Morehopqa: More than multi-hop reasoning. *arXiv preprint arXiv:2406.13397*, 2024.
- Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. *arXiv preprint arXiv:2409.10502*, 2024.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.
- Dimos Tsouros, H el ene Verhaeghe, Serdar Kadio glu, and Tias Guns. Holy grail 2.0: From natural language to constraint models. *arXiv preprint arXiv:2308.01589*, 2023.
- Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.
- Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can’t plan; can lrms? a preliminary evaluation of openai’s o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024.
- Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models, 2024a. URL <https://arxiv.org/abs/2406.14852>.
- Kevin Wang, Junbo Li, Neel P Bhatt, Yihan Xi, Qiang Liu, Ufuk Topcu, and Zhangyang Wang. On the planning abilities of openai’s o1 models: Feasibility, optimality, and generalizability. *arXiv preprint arXiv:2409.19924*, 2024b.
- Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *ACL*, 2024c.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*, 2024.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.
- Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.

Appendix

A Detailed Evaluation Settings

We first present a detailed list of various object and relation numbers, together with other settings for each task.

A.1 Plan Generation

The combination of the number of objects and the number of relations for Plan Generation task:

**Plan Generation data groups:
NumObjects_NumRelations**

3_3,
5_5, 5_7, 5_9,
10_10, 10_15, 10_20, 10_25, 10_30, 10_35, 10_40, 10_45,
15_15, 15_24, 15_34, 15_43, 15_53, 15_62, 15_72, 15_81, 15_91, 15_100,
20_20, 20_29, 20_38, 20_47, 20_56, 20_65, 20_74,
20_83, 20_92, 20_101, 20_109, 20_118, 20_127, 20_136,
20_145, 20_154, 20_163, 20_172, 20_181, 20_190,
25_25, 25_34, 25_43, 25_53, 25_62, 25_71, 25_80, 25_89,
25_99, 25_108, 25_117, 25_126, 25_136, 25_145, 25_154,
25_163, 25_172, 25_182, 25_191, 25_200,
30_30, 30_39, 30_48, 30_57, 30_66, 30_75, 30_84, 30_93,
30_102, 30_111, 30_119, 30_128, 30_137, 30_146, 30_155,
30_164, 30_173, 30_182, 30_191, 30_200,
40_40, 40_49, 40_58, 40_67, 40_76, 40_85, 40_94, 40_103,
40_112, 40_121, 40_130, 40_139, 40_148, 40_157, 40_166,
40_174, 40_183, 40_192, 40_201, 40_210, 40_219, 40_228,
40_237, 40_246, 40_255, 40_264, 40_273, 40_282, 40_291, 40_300,
50_50, 50_59, 50_67, 50_76, 50_84, 50_93, 50_102, 50_110,
50_119, 50_128, 50_136, 50_145, 50_153, 50_162, 50_171,
50_179, 50_188, 50_197, 50_205, 50_214, 50_222, 50_231,
50_240, 50_248, 50_257, 50_266, 50_274, 50_283, 50_291, 50_300

The prompt template we use with these groups in task Plan Generation:

Plan Generation task prompt template

You are tasked with generating a list of logical and consistent relationships between objects:

- Each object should have a unique name.
- The relationships use either the "<" (smaller than) or ">" (larger than) operator.
- The relationships must be logically consistent and acyclic (i.e., no cycles like $A > B > A$).
- Not all objects need to be directly related, but all relationships must be valid.

Now generate a logical and consistent list of `{{num_relations}}` relationships with `{{num_objects}}` objects.

Output Format:

Return only a valid JSON object after the <RESPONSE> tag, structured as follows:

```
<RESPONSE>
{
  "relationships": [
    "<object> <operator> <object>",
    "<object> <operator> <object>",
    ...
    "<object> <operator> <object>"
  ]
}
</RESPONSE>
```

A.2 Consistency Detection

We provide the groups of the number of objects, the number of relations, and the minimum cycle length of the directed graph. We generate 20 unique graphs for each group.

Consistency Detection data groups: NumObjects_NumRelations_MinCycleLength

```
10_15_0, 10_15_3, 10_15_6,
10_30_0, 10_30_3, 10_30_6,
20_30_0, 20_30_6, 20_30_12,
20_60_0, 20_60_6, 20_60_12,
30_45_0, 30_45_6, 30_45_12, 30_45_24,
30_90_0, 30_90_6, 30_90_12, 30_90_24
```

The prompt template we use in task Consistency Detection:

**Comparison Question data groups:
NumObjects_NumRelations_DepthofComparison**

10_15_0, 10_15_2, 10_15_4, 10_15_6,
10_30_0, 10_30_2, 10_30_4,
20_30_0, 20_30_2, 20_30_4, 20_30_6,
20_60_0, 20_60_2, 20_60_4, 20_60_6,
30_45_0, 30_45_2, 30_45_4, 30_45_6,
30_90_0, 30_90_2, 30_90_4, 30_90_6

The prompt template we use in the task Comparison Question:

Comparison Question task prompt template

Given a set of relationship statements:
{{relationships}}

And a comparison statement to evaluate:
{{comparison}}

Your task is to determine if the comparison is True, False,
or Unknown based solely on the provided relationships.

Guidelines for evaluation:

- True: The comparison can be logically deduced from the given relationships
- False: The comparison can be proven incorrect using the given relationships
- Unknown: There is insufficient information in the relationships to definitively prove the comparison true or false

Important notes:

- Use ONLY the information provided in the relationships
- Do not make assumptions beyond what is explicitly stated
- If there are multiple possible interpretations, mark as Unknown
- You can do whatever analysis that you need in the response, but your response must end with the OUTPUT format below
- Do not add any additional text after the output

Required output format:

OUTPUT:

True/False/Unknown

Comparison Question task self-correction prompt template

Are you sure?

Note: You can do whatever analysis that you need in the response,
but your response must end with the OUTPUT format below.

Do not add any additional text after the output.

Required output format:

OUTPUT:

True/False/Unknown

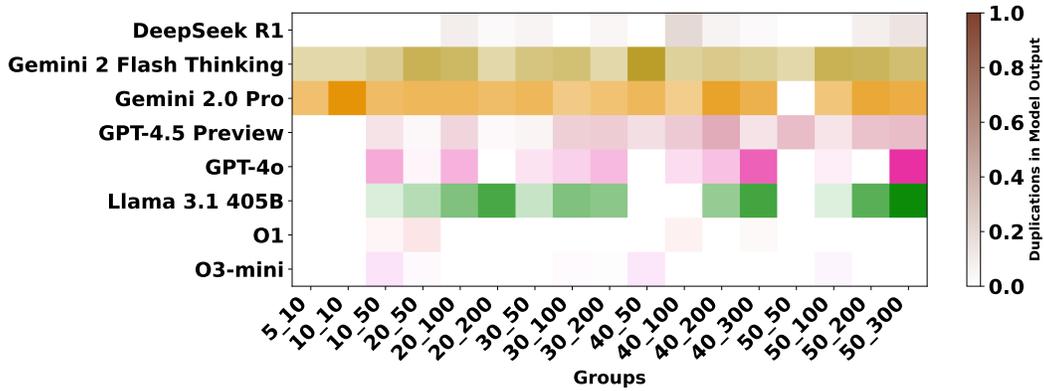


Figure 7: Plan Generation Relation Duplications in Model Output

B Detailed Task Results

B.1 Detailed Plan Generation Task Results

Figure 10, Figure 11, Figure 12 and Figure 13 show the detailed result of Plan Generation task. The rows are the number of objects, and the columns are the number of relations prompted to the model. The empty cells are the combinations that are not covered in this test. Each colored cell shows the average accuracy result of 3 runs.

We also notice that some models are easier to output duplicated relations in general, like Gemini models. And some models get easier to output duplicate relations as the task gets harder, such as GPT-4o and Llama 3.1 405B. Figure 7 shows the detailed results of the average existence of duplications (True of False) in model's output for each model and task groups.

B.2 Detailed Consistency Detection Task Result

Figure 8 shows the detailed result for Consistency Detection task. The average F1 score is aggregated based on the group of the number of objects, the number of relations and the minimum cycle length in the provided relations to the model. We performed 5 runs for each model, and calculated the average F1 score.

B.3 Detailed Comparison Question Task Results

Figure 9 shows the detailed result for Comparison Question task. The average accuracy is aggregated based on the group of the number of objects, the number of relations and the depth of the provided comparison statement to the model. Except for Claude 3.7 Sonnet, We performed 5 runs for each model, and calculated the average.

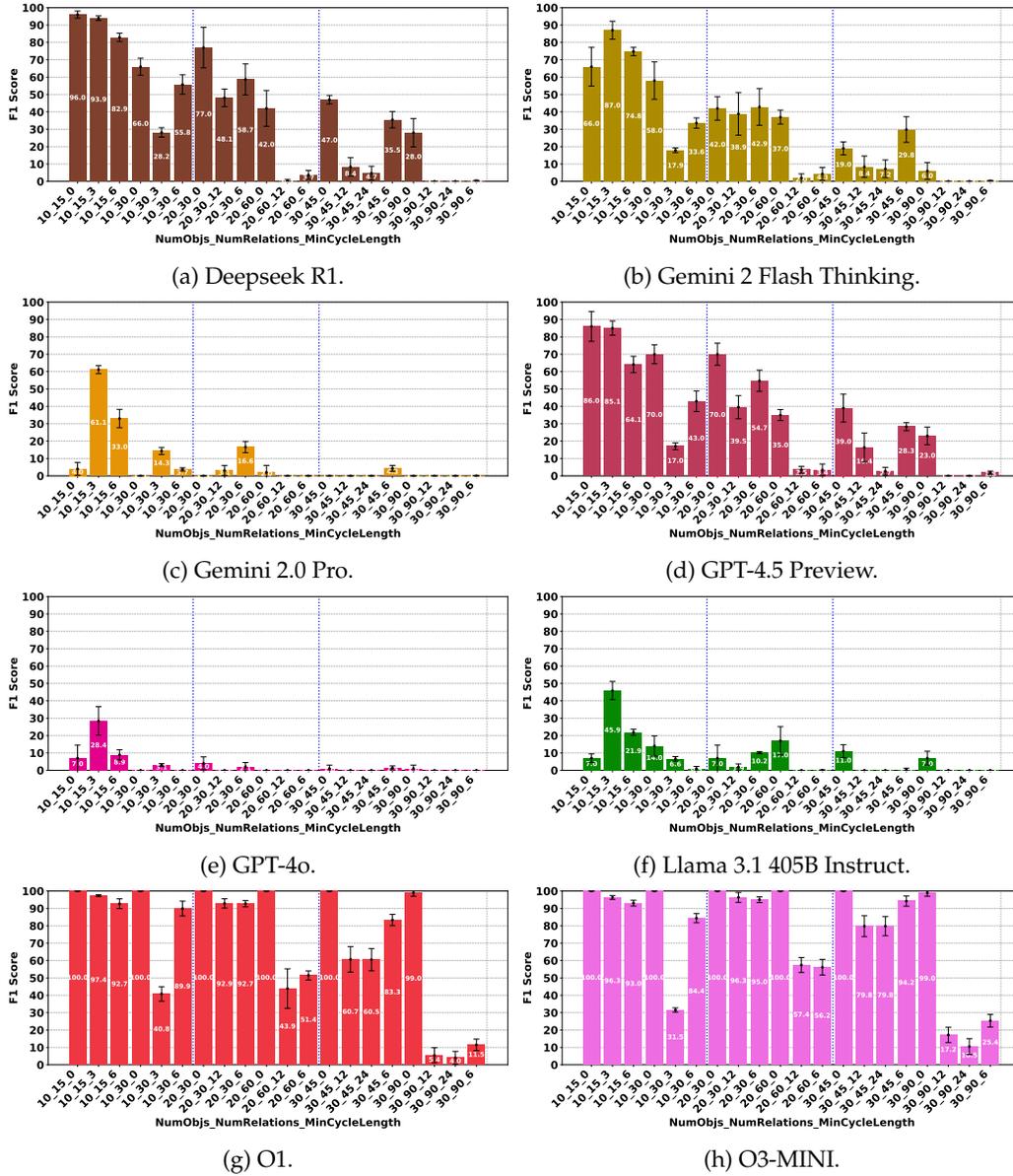


Figure 8: Detailed Average F1 Score of Consistency Detection Task - Continued

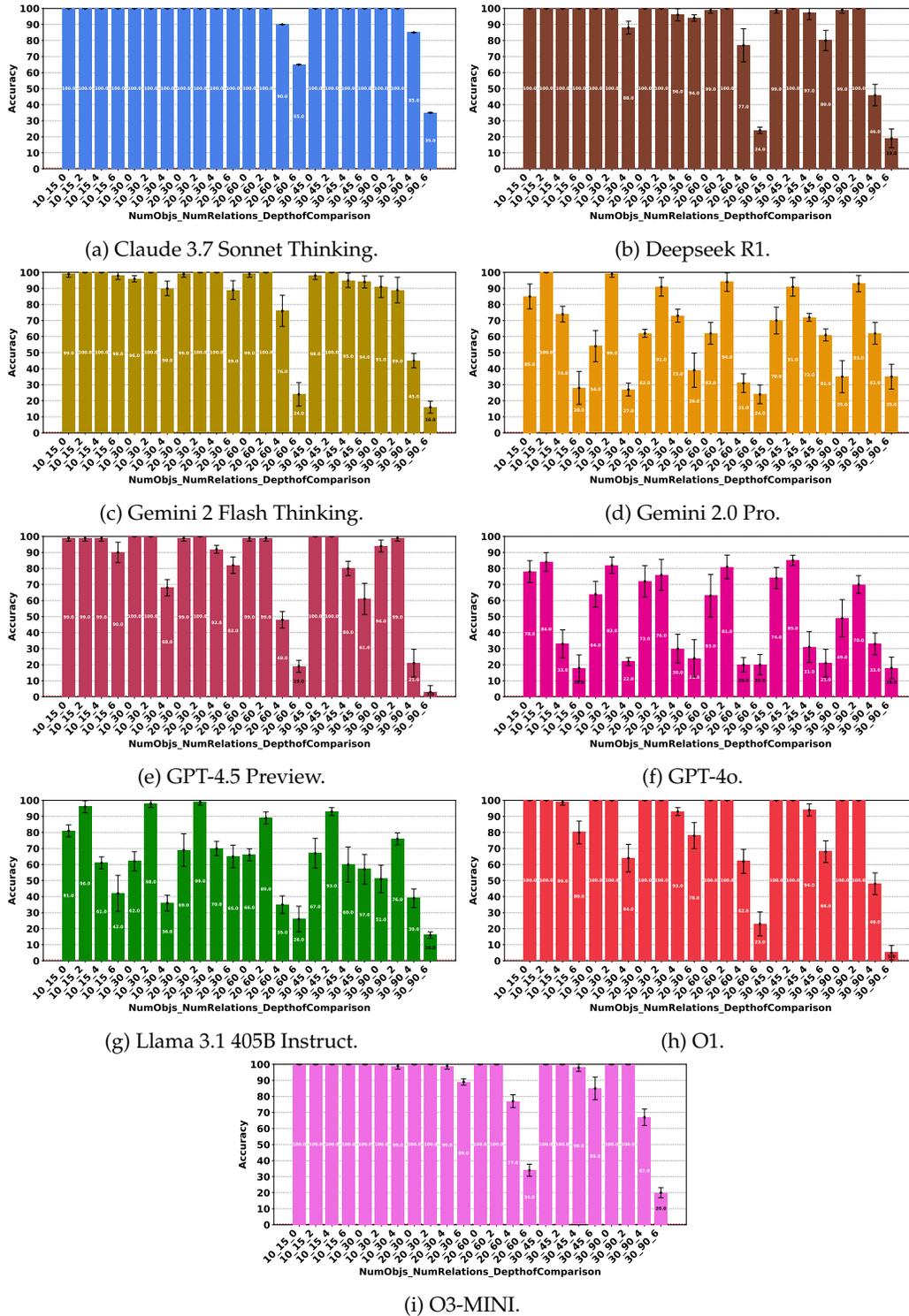
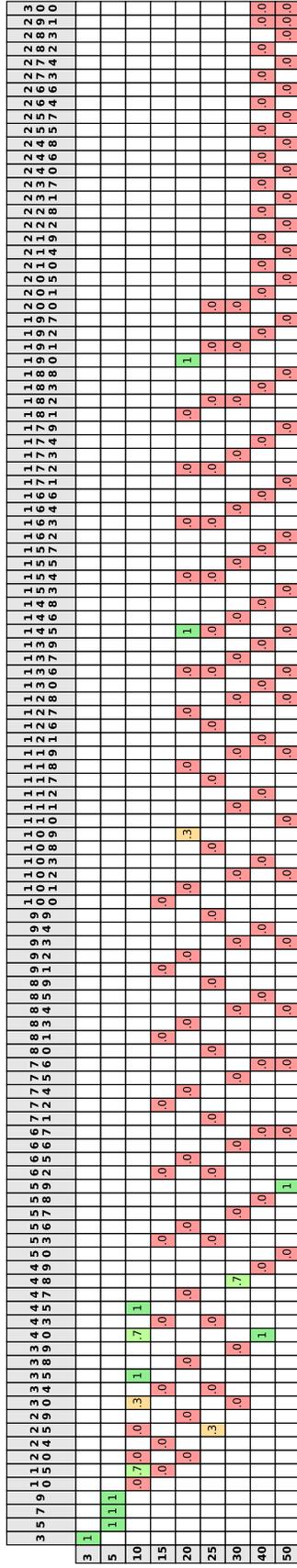
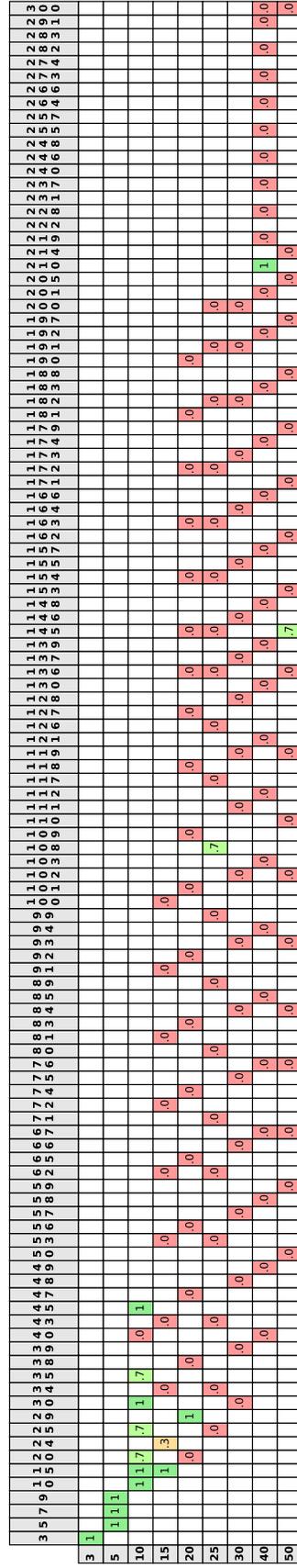


Figure 9: Detailed Average Accuracy of Comparison Question Task



(a) GPT-4o.



(b) Llama 3.1 405B Instruct.

Figure 12: Detailed average accuracy of Plan Generation task - Continued

