A ∧ B ⇔ B ∧ A: Evaluating and Improving the Logical Reasoning Ability of Large Language Models

Anonymous ACL submission

Abstract

We introduce LogicAsker, a novel approach for evaluating and enhancing the logical reasoning capabilities of large language models (LLMs) such as ChatGPT and GPT-4. Despite their prowess in tasks like writing assistance, code generation, and machine translation, assessing LLMs' ability to reason has been challenging. Traditional evaluations often prioritize accuracy on downstream tasks over direct assessments of reasoning processes. LogicAsker addresses this gap by employing a set of atomic reasoning skills grounded in propositional and predicate logic to systematically examine and improve the reasoning prowess of LLMs. Our methodology reveals significant gaps in LLMs' learning of logical rules, with identified reasoning failures ranging from 25% to 94% across different models. Moreover, we leverage these findings to construct targeted demonstration examples for in-context learning, notably enhancing logical reasoning in models like GPT-4 by up to 10%. To our knowledge, this is the first effort to utilize test case outcomes to effectively refine LLMs' formal reasoning capabilities. We will make our code, data, and results publicly available to facilitate further research and replication of our findings.

1 Introduction

Large language models (LLMs), such as OpenAI's GPT series have significantly impacted natural language processing, excelling in a variety of tasks including text generation, machine translation, and code generation (Gao et al., 2022, 2023a; Jiao et al., 2023). Notably, ChatGPT has achieved rapid adoption, reaching 100 million users in just two months (Hu, 2023). Despite their success, the true reasoning capabilities of these models remain under scrutiny.

Reasoning, defined as the cognitive process of using logic to draw conclusions from given facts (Wei et al., 2022b,a), is crucial for complex interactions that go beyond straightforward text generation. Accurately assessing this ability in LLMs is essential, yet challenging, as models may correctly perform tasks merely relying on shortcuts such as pattern recognition without truly engaging in logical reasoning (Huang and Chang, 2022; Huang et al., 2023; Liu et al., 2023). We provide a motivating example in Appendix A.

To better handle these challenges, a well-performing testing framework should be able to define a set of skills that **a) directly correspond to the reasoning process, b) cannot be further divided, c) cover all formal logical reasoning scenarios, and d) can identify LLMs' weaknesses and facilitate improving LLMs' performance.** Property a) ensures that the task cannot be accomplished by other approaches, such as inferring from the correlations of words, and the evaluation result directly reflects the model's reasoning ability. Property b) and c) ensure that the set of skills is fundamental and comprehensive, which can provide helpful insights to accomplish Property d).

Based on these criteria, we propose LogicAsker, an automatic framework to evaluate and improve LLMs' formal reasoning ability on a set of atomic skills. We adopted the concept of Minimum Functionality Tests (MFTs) (Ribeiro et al., 2020), which are analogous to unit tests in software engineering, where a collection of simple examples is used to check a specific behavior within a capability. The tests are particularly useful for detecting when models use shortcuts to handle complex inputs without actually mastering the capability(Ribeiro et al., 2020). Specifically, we first construct the set of atomic skills by collecting and combining all basic principles and laws in propositional and predicate logic, two fundamental systems used to formalize reasoning procedures (Partee et al., 1990), together with a set of common logical fallacies (Hurley and Watson, 2020). Based on the skill set, LogicAsker systematically generates reasoning questions by converting standard logic expressions into nat-

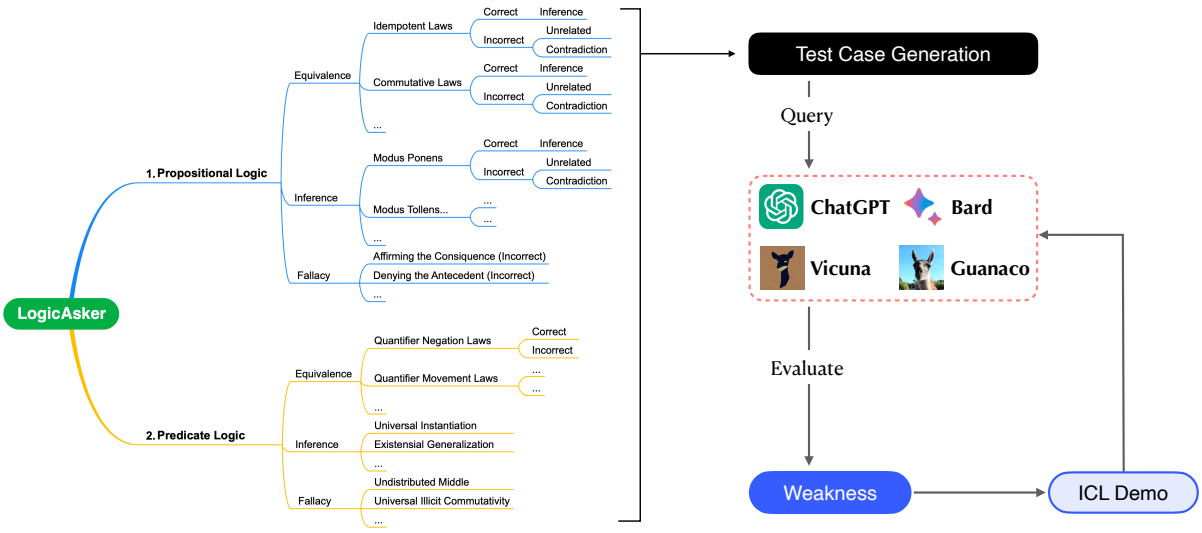

Figure 1: Overview of the LogicAsker framework.

ural languages. According to the questions and answers, LogicAsker calculates the LLM's accuracy on each skill, identifies the weaknesses of the LLM, and finally generates demonstration examples to improve the LLM's reasoning capacity using in-context-learning techniques (Brown et al., 2020). In addition, for a single skill, LogicAsker utilizes a wide range of vocabulary to translate it into various natural language queries and calculate the average performance over all queries, avoiding the result being affected by word correlations in the sentence.

Table 1 compares our framework to previous studies, which provide datasets for testing the reasoning ability of models. As seen, many of these datasets are not amenable and thus vulnerable to data leakage issues, i.e., can be memorized or exploited by LLMs trained on the massive corpora from the Internet. Other programmable datasets are of limited scope. In contrast, our framework is the most comprehensive one and also the only one that can utilize the evaluation result to improve LLMs' reasoning abilities.

To assess the performance of LogicAsker, we conducted comprehensive testing on six widely deployed LLMs, including four commercial LLMs (GPT-3, ChatGPT, GPT-4, and Google Bard) and two open-source LLMs (Vicuna and Guanaco) four of which are ranked within the top 8 in the LLM Arena Leaderboard proposed by (Zheng et al., 2023).

The results demonstrate that the test cases generated by LogicAsker effectively identified logical reasoning failures in different commercial LLMs and research models at a rate (i.e., 1 − accuracy) ranging from 25% to 94%. Furthermore, the test cases generated by LogicAsker can be utilized to design demonstration examples for in-context learning, improving LLMs' logical reasoning abilities. For example, in the case of GPT-4, applying in-context learning using LogicAsker's test cases resulted in a substantial enhancement, improving the logical reasoning ability from 75% to 85%. All the code, data, and results will be released for reproduction and future research.[1].

We summarize the main contributions of this work as follows:

- We are the first work that formally defines a set of 30 atomic skills and 208 extended skills that an LLM should possess to perform formal reasoning based on propositional logic and predicate logic, two fundamental systems of formal logic.
- We develop LogicAsker, a fully automatic tool that can generate test cases under the basic skills and provide insights into LLMs' reasoning capacities, and we are the first work that can create prompts based on testing results to improve the performance of LLMs effectively.
- We perform a comprehensive empirical evaluation of six widely deployed LLMs based on logical reasoning ability. We demonstrate that the test results by LogicAsker can be used to effectively evaluate and improve the performance of LLMs.

[1] https://drive.google.com/drive/folders/19xj5XjnSbtlY1vvT0kbcKfY1FfvCnE9j?usp=share_link

Table 1: Comparison with previous works.

	Fully Automatic	Atomic Skills	Formal Rules	Include Fallacies	Identify Weakness	Improve LLMs	LLMs* Tested	Example Testbed
CLUTRR (Sinha et al., 2019)	×	×	×	×	✓	×	-	BERT
LogiQA (Liu et al., 2020)	×	×	×	×	×	×	-	BERT
RECLOR (Yu et al., 2020)	×	×	×	×	✓	×	2	GPT2
Soft Reasoner (Clark et al., 2020)	✓	×	1	×	✓	×	-	RoBERTa
LogicNLI (Tian et al., 2021)	×	×	7	×	✓	×	-	BERT
FOLIO (Han et al., 2022)	×	×	×	×	×	×	4	GPT3
LogicInference (Ontañón et al., 2022)	✓	×	19	×	×	×	-	T5
ProntoQA-OOD (Saparov et al., 2023)	✓	×	6	×	✓	×	4	GPT3.5
LogicAsker (Ours)	✓	✓	30	✓	✓	✓	6	GPT4

* We consider language models with more than 1 billion parameters as LLMs.

2 Preliminaries

2.1 Formal Analysis of Reasoning Abilities

"Reasoning" can be characterized into formal reasoning and informal reasoning. The former is a systematic and logical process that follows a set of rules and principles, and the reasoning within these systems will provide valid results as long as one follows the defined rules (e.g., all A are B, all B are C; therefore, all A are C). The latter is a less structured approach that relies on intuition, experience, and common sense to draw conclusions and solve problems (Huang and Chang, 2022; Bronkhorst et al., 2020) (e.g., Hong Kong residents have a high life expectancy; this is probably because they have healthy living habits). Generally, formal reasoning is more structured and reliable, whereas informal reasoning is more adaptable and open-ended but may be less reliable. In this paper, we focus on the formal reasoning process to systematically analyze LLMs' reasoning abilities.

To formalize reasoning procedures, two fundamental systems are usually adopted, namely, propositional logic and predicate logic. The former one deals with propositions or statements that can be either true or false, and utilizes logical operators including $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (inference), and $\leftrightarrow$ (bidirectional) to connect these statements. The latter one, in contrast, extends propositional logic to deal with more complex statements that involve variables, quantifiers, and predicates. Both propositional logic and predicate logic contain various rules for the reasoning process. These rules can be categorized into equivalence rules and inference rules. Equivalent rules summarize the basic expressions that are equivalent in terms of truth value (e.g., $\neg(P \wedge Q) \Leftrightarrow (\neg P) \vee (\neg Q)$). Inference rules summarize the basic valid inference rules (e.g., from the premises: $A \rightarrow B$, and A, we can infer B).

We refer to (Partee et al., 1990) for a more detailed explanation. Table 7-9 in Appendix B list common inference rules in predicate logic and propositional logic. Besides inference rules, formal logic systems can also express common logical fallacies, i.e., arguments that may sound convincing but are based on faulty logic and are, therefore, invalid. We list the common logical fallacies in Table 10.

2.2 Minimum Functionality Test

In this paper, we adopted the concept of Minimum Functionality Tests (MFTs), introduced in (Ribeiro et al., 2020), to evaluate the reasoning ability of LLMs. MFTs are analogous to unit tests in software engineering, where a collection of simple examples is used to check a specific behavior within a capability. These tests involve creating small and focused datasets that are particularly effective in detecting whether models resort to shortcuts to handle complex inputs, rather than truly mastering the capability.

To apply MFTs in evaluating the reasoning ability of LLMs, we treated each formal logical rule as an independent task and generated abundant test cases for each task. Each test case was designed to trigger logical failures in the LLMs, allowing us to assess the strengths and weaknesses of LLMs in the logical reasoning process, and providing a solid foundation for further analysis and improvement.

3 LogicAsker

In this section, we introduce the design and implementation of LogicAsker, a novel tool to trigger logical reasoning failures in large language models. Figure 1 overviews the workflow of LogicAsker, which consists of three main modules: test case generation, weakness identification and in-context learning (ICL) demonstration. In particular, the test case generation module utilizes atomic skills

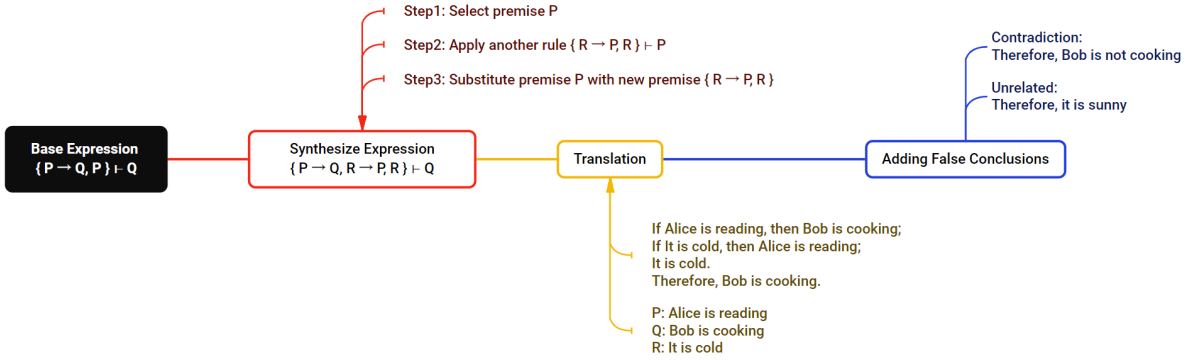

Figure 2: Test case generation procedure.

defined on the two formal logic systems and an inference synthesis approach to generate questions as test cases. Then, the generated cases are fed into the LLMs to reveal weaknesses and provide insights into the LLMs by the weakness identification process. Finally, LogicAsker utilizes these insights to construct ICL demonstrations to improve the reasoning abilities of the LLMs.

3.1 Reasoning Skills

Atomic skills. As described in Section 2.1, propositional and predicate logic are two fundamental systems that formalize the reasoning process. The inference rules and equivalence laws in these two systems are atomic and can cover all correct reasoning scenarios; therefore, we define these 30 rules as the set of atomic skills an LLM should possess to perform formal reasoning.

Extended skills. Predicate logic extends propositional logic to deal with more complex statements that involve variables, quantifiers, and predicates. In this regard, besides the unique equivalence and inference laws in predicate logic, we add quantifiers and variables to every rule in propositional logic to form the predicate version of the laws. Using this approach, we expand the set of 30 atomic skills into a set of 208 extended skills. In Appendix C, we provide some concrete examples of these extended rules.

3.2 Test Case Generation

To generate logical questions, LogicAsker first adopts a rule-based method to generate logical expressions systematically based on reasoning skills and then translates the logical expressions into natural language. Figure 2 provides an overview of the procedure.

Logic expression generation. To better control the process of logic expression generation, we first define the length of an inference problem by the number of syllogisms it involves. We use the inference rules described in Section 2.1 to generate inference expressions with length one. When a longer inference (> 1) is specified, we start with a base expression $E_0 := P_1 \wedge P_2 \rightarrow C_1$ with length one and expand the inference chain. Specifically, we substitute the premises (either or both) of the first inference with the conclusion of some other syllogism and append the premises of those syllogisms into the list of all premises. For example, we can find another syllogism $E_1 := P_3 \wedge P_4 \rightarrow P_2$ with P_2 as the conclusion and then obtain a new expression $E_{new} := P_1 \wedge P_3 \wedge P_4 \rightarrow C_1$ with the inference length of two. We can obtain inference expressions of any length by recursively expanding the inference chain as above. During the generation process, one can specify the desired rules and length to allow complete control over expected test cases.

In addition to the correct inference expression created above, we generate three kinds of false inference expressions: contradiction, unrelated, and fallacy. A contradiction is generated by negating the conclusion of a correct inference expression and an unrelated is generated by replacing the conclusion of a valid inference expression with an irrelevant statement. For example, for $E_0 := P_1 \wedge P_2 \rightarrow C_1$, a contradiction is $E_c := P_1 \wedge P_2 \rightarrow \neg C_1$, an unrelated can be $E_u := P_1 \wedge P_2 \rightarrow U_1$. We create a fallacy by directly using the fallacy rules listed in Section 2.1 for an inference length of one. For a fallacy with a more extended length, we select a fallacy rule as the base expression and expand the inference chain using correct rules, ensuring the expression's incorrectness.

Natural language translation. Partially inspired by (Ontañón et al., 2022), translating a clause into natural language involves a series of patterns that depend on the structure of the clause.

Simple propositions are transformed into one of the template patterns, such as “subject verb-action”, “subject predicate”, or “impersonal-action” with a predefined set of subjects, verbs, predicates, and impersonal actions that can be chosen randomly without repetition. For predicate clauses that involve constant or variables, we employ template “subject verb-action”, “subject predicate” to translated them. Furthermore, each clause can be rendered in various modes, such as the present, past, or negated forms. Additionally, connectives like "or," "and," "implies," and "if and only if" also adhere to their designated patterns. For quantified clauses, we adopt patterns like "for all x, X", "there is at least one x for which X", and "some Xs are Y,". To facilitate the generation process, we curate extensive lists of potential subjects, including common names in English, and compile plausible predicates, actions, and impersonal actions. We provide a detailed illustration of the translation process in Appendix D.

3.3 Weakness Identification

Generally, LLMs are required to perform well on two tasks to respond appropriately to a query involving reasoning, i.e., instruction following and logical reasoning. The former ensures LLMs can understand the instructions in the query and respond as required. At the same time, the latter makes sure LLMs can successfully resolve the problem through reasoning.

To measure the reasoning abilities of the LLMs, we define the response accuracy as follows. Let N_{satisfy} denote the number of responses that satisfy the requirement in the query (instruction following), and N_{correct} denote the number of responses that are correct (reasoning). In particular, since all generated queries are formulated as yes-or-no questions, LogicAsker adopts an automatic approach that searches for pre-defined keywords (e.g., "yes" and "no") in sentences to identify qualified answers and correct answers. The response accuracy is then calculated by

$$\text{Response Acc} = \frac{N_{\text{correct}}}{N_{\text{satisfy}}}.$$

This metric can directly reflect LLMs' performance on reasoning, ruling out the instruction following factor.

To reveal the weaknesses of LLMs, we generate n test cases for each leaf node in the rule tree depicted in Figure 1. Then, we calculated the response accuracy of an LLM of each leaf node. Based on the result, we can identify the weaknesses of LLMs by listing the leaf nodes that receive the lowest accuracy. In addition, by grouping the accuracy by different attributes in the rule tree, we can gain insights into the strengths and weaknesses of LLMs on these attributes (e.g., performance on predicate logic vs. propositional logic).

3.4 Improving LLMs

In-context learning (ICL) is a paradigm that enables LLMs to learn tasks with examples in the form of demonstrations (Brown et al., 2020). It leverages task instructions and a few demonstration examples to convey the task semantics, which are then combined with query questions to create inputs for the language model to make predictions. ICL has demonstrated impressive performance in various natural language processing and code intelligence. However, the performance of ICL is known to rely on high-quality demonstrations (Gao et al., 2023b) strongly. To fully unleash the potential of ICL, LogicAsker utilizes the weak skills of each LLM to construct both correct and incorrect examples with expected answers and explanations as demonstrations to facilitate the reasoning of LLMs. The generation process follows a similar approach to the test case generation described in § 3.2, with the difference being that we append the correct answer and a brief explanation at the end of each case. We show an instance of the demonstration example and generation process in Appendix E.

4 Experiments

4.1 Experimental Setup

We apply LogicAsker to test six popular LLMs, including four from commercial companies and two from open-source. Table 2 lists brief information on these systems. Among them, four LLMs are ranked within the top 8 in the LLM Arena Leaderboard proposed by (Zheng et al., 2023), according to the assessment results in June 2023. We leave details of how we access the model, the parameters used, and the prompt we used in Appendix F.

4.2 Effectiveness of LogicAsker

We demonstrate the effectiveness of LogicAsker through the overall performance of LLMs on the test cases. We conduct two iterations of experiments for a comprehensive assessment. In the first iteration, we follow the setting in § 3.3 and set

Table 2: Conversational LLMs used in the evaluation.

Name	Organization	Launch Date	Rank
GPT-4	OpenAI	Mar 2023	1
ChatGPT	OpenAI	Nov 2022	4
GPT-3 (Brown et al., 2020)	OpenAI	Jun 2020	-
Bard	Google	Mar 2023	-
Vicuna-13b	LMSYS Org	Mar 2023	6
Guanaco-33b (Dettmers et al., 2023)	UW	May 2023	8

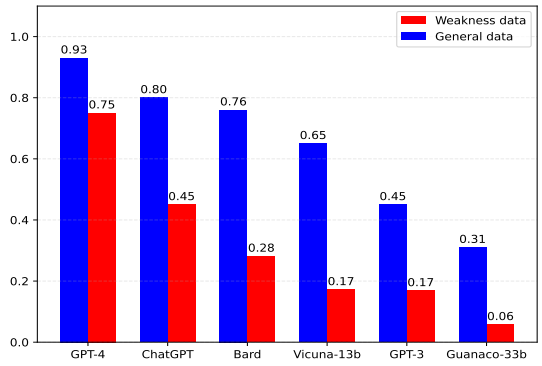

Figure 3: Overall accuracy.

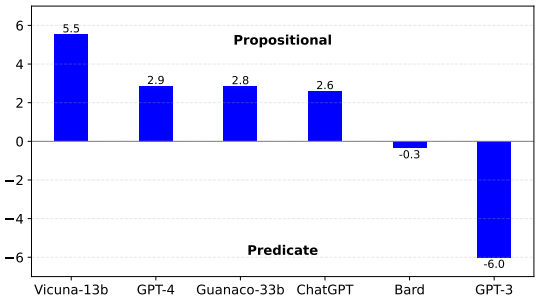

Figure 4: Propositional minus predicate accuracy (%).

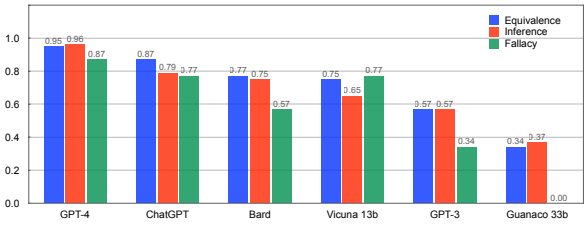

Figure 5: Accuracy of different rule categories.

$n = 10$, resulting in 2080 cases. The second iteration is based on the first one, which focuses on the identified weaknesses of each LLM, i.e., the ten leaf nodes in Figure 1 with the lowest accuracy. We generated ten additional test cases for each weakness. These 100 test cases comprise our "weakness dataset, " which will be utilized for further evaluation in 4.5.

The overall performance of LLMs in the first and second iteration is shown in Figure 3. The result reveals that our framework can effectively expose logical failures in the first iteration, with LLM's accuracy ranging from 31%-93%. When focusing on the weak skills of LLMs in the second iteration, we further reduce the accuracy to 6%-75% for the LLMs. What's surprising is that most of these LLMs achieved response accuracy even lower than random guesses (i.e., 50% here) when confronted with logical questions involving specific logical rules. This contradicts their remarkable performance in various LLM benchmarks, for example, achieving top 8 ranks on the LLM Arena Leaderboard. It suggests that existing benchmark datasets are not comprehensive enough to assess the generalization ability of LLMs in reasoning.

4.3 Insights into Reasoning Abilities

We conducted a comprehensive analysis to gain insights from the failures exposed by LogicAsker, obtaining three key observations from the evaluation:

Most LLMs are better at easier logical skills. We compared the performance of LLMs on propositional logic and predicate logic, the former of which is simper in form while the latter involves more complex quantifier manipulations. Figure 4 illustrates the difference between the accuracy and response scores obtained for the two logic systems. A positive value indicates a higher score in propositional logic, while a negative value indicates higher scores in predicate logic. Notably, we observed that most LLMs are better at propositional logic, implying their limited ability in complex reasoning scenarios.

Most LLMs are weak in recognizing logical fallacies. Figure 5 presents the accuracy of LLMs under different skill categories. Interestingly, we discovered that among three types of skills, recognizing fallacies has the lowest accuracy for most LLMs, with Vicuna-13b being the only exception. Particularly, Guanaco 33b achieved zero accuracy on the fallacy task due to its tendency to provide affirmative answers to most queries. It suggests

that current LLMs are over-confident even in fallacies, which may be learned from the mistakes in pretraining data.

Longer inference chains are more challenging. To assess the impact of inference length, we generated test cases of varying lengths (i.e., ranging from 1 to 7) using randomly selected rules. For each length, we generated 100 test cases. Table 3 shows the performance of LLMs on these test cases. Generally, most LLMs perform gradually worse as the inference length increases, indicating the increased complexity introduced by longer inference chains. Particularly, Guanaco 33b suffers from a severe prediction bias such that it tends to output affirmative answers to all questions, regardless of the inference length or logical complexity.

Table 3: Accuracy with respect to inference length.

Length	1	3	5	7
GPT-4	0.92	0.85	0.78	0.74
ChatGPT	0.79	0.71	0.72	0.65
Bard	0.80	0.68	0.63	0.56
Vicuna 13b	0.63	0.62	0.52	0.48
GPT-3	0.68	0.52	0.60	0.56
Guanaco 33b	0.57	0.40	0.46	0.55

Case study: GPT-4 did not learn all logic rules well. To provide a direct impression of what skills LLMs cannot perform well, we list three atomic rules in which GPT-4 has the lowest accuracy in Table 4. While GPT-4 has an average accuracy of 93% over all skills, it only achieves 60% - 70% accuracy on these skills, indicating that it cannot perform these atomic skills smoothly.

These insights provide a valuable understanding of the strengths and weaknesses of each LLM when handling logical questions, allowing us to uncover specific areas that require improvement and potential avenues for enhancing overall performance.

4.4 Validity of Test Cases

In this section, we aimed to investigate the validity of the test cases generated by LogicAsker. To achieve this, we randomly sampled 10% (208) of the test cases generated during the first iteration of the experiment in 4.2 and conduct manual inspection. Two annotators with bachelor's degrees were recruited to answer the questions manually. Each test case was annotated as either valid or invalid based on the following three questions: **a)** Is the question grammatically correct? **b)** Is the question understandable and has only one interpretation? **c)** Can the target answer be derived from the question? A test case is considered valid only when both annotator's answer to the above questions are negative. The results of the annotation are presented in Table 5. This result is statistically sufficient to prove that the probability of LogicAsker generating understandable and solvable logical questions is larger than or equal to 0.94 (with p-value 0.05), indicating that the queries created by LogicAsker are highly reliable and valid.

4.5 LogicAsker to Improve Reasoning

In this section, we explore the potential of LogicAsker in further improving the reasoning ability of LLMs through in-context learning (ICL).

We employ LogicAsker to generate ICL demonstrations tailored to address the weaknesses dataset uncovered in the experiments of 4.2. For each inference problem, we generated ICL demonstrations that provide both the expected answer and an explanation as described in § 3. We evaluate the effectiveness of the ICL demonstrations generated by LogicAsker by comparing the following prompting strategies: a) Zero-Shot: We provide only task instructions without any ICL demonstrations. b) Random Demonstrations: In addtion to the task instruction, we also include four ICL demonstrations selected randomly from the available rules. c) Weakness ICL Demonstration: Instead of random demonstrations, we include four ICL demonstrations using the weakness rules identified in 4.2 with balanced answer labels, i.e., two correct and two incorrect.

We perform ICL with the GPT family on their respective weakness datasets and report the results in Table 6. In general, the weakness ICL demonstrations are more effective than those random ICL demonstrations. Though the latter one perform slightly better on ChatGPT, it brings no improvement to GPT-3. These findings demonstrate the potential of LogicAsker in improving the reasoning ability of LLMs.

5 Related Work

Numerous recent studies have attempted to measure the reasoning ability of LLMs. One approach to gauge the reasoning abilities of LLMs is by assessing their performance, such as accuracy, on tasks that demand reasoning skills, including arithmetic reasoning (Cobbe et al., 2021; Hendrycks et al., 2021; Amini et al., 2019; Patel

Table 4: Weakness of GPT-4

Rule	Type	Example	Accuracy
De Morgan's laws	Correct	Jessica is making tea and it is overcast cannot both be true. Therefore, Jessica is not making tea or it is not overcast.	0.6
Conditional laws	Incorrect	Karen is not playing a game or it is sunny. Therefore, the fact that Karen plays a game does not imply that it is sunny.	0.6
Biconditional introduction	Incorrect	If Tom writes letters, then Bob is running. If Bob runs, then Tom is writing letters. Therefore, it is not true that Bob is running if and only if Tom is writing letters.	0.7

Table 5: Validity of test cases.

Invalid Cases	a	b	c	Total
Count	4	3	0	7
Percentage	1.92%	1.44%	0.00%	3.37%

Table 6: Performance of ICL demonstrations by LogicAsker.

Models	Zero	Random	Weak
GPT-4	0.75	0.83	**0.85**
ChatGPT	0.45	**0.64**	0.56
GPT-3	0.17	0.16	**0.39**

et al., 2021; Miao et al., 2020; Ling et al., 2017; Roy and Roth, 2016), commonsense reasoning (Talmor et al., 2019; Geva et al., 2021; Clark et al., 2018), symbolic reasoning (Wei et al., 2022b), understanding of words, dates, and causal relationships (Aarohi Srivastava, 2022), generalization ability (Lake and Baroni, 2017; Anil et al., 2022), and table reasoning ability (Nan et al., 2021). However, whether LLMs' predictions are based on true reasoning or simple heuristics remains unclear, as most existing evaluations focus solely on accuracy on end tasks and do not directly assess their reasoning processes.

There have also been efforts to develop metrics and benchmarks that enable a more formal analysis of reasoning in LLMs. For instance, (Han et al., 2022) use expert-written data to create a dataset that contains first-order logic reasoning problems, requiring models to determine the correctness of conclusions given a set of premises. Similarly, (Saparov and He, 2022) utilizes one predicate inference rule recursively to generate test cases, while (Ontañón et al., 2022) adopts mainly propositional logic rules and a randomized generation method to synthesize logical expressions as test cases. Nonetheless, these methods either lack generalizability or focus on a limited set of deduction rules. Recently, (Saparov et al., 2023) proposed a method to evaluate LLMs' general deductive reasoning capacity by employing all deduction rules in propositional logic, measuring their ability to generalize to more complex proofs than their demonstrations. In contrast, our work encompasses a broader scope, considering all deduction rules and equivalent laws in propositional logic and predicate logic, the two fundamental formal logic systems. Additionally, our framework is designed to provide a comprehensive insight into the models' capacity for each particular rule and employ this insight to enhance LLMs' performance.

6 Conclusion

In this paper, we present LogicAsker, an automated tool designed to comprehensively evaluate and improve the formal reasoning abilities of LLMs under a set of atomic skills.

Our research demonstrated the efficacy of LogicAsker in identifying logical reasoning failures in a diverse set of widely deployed LLMs, we achieved a substantial success rate in revealing reasoning flaws in these models, ranging from 25% to 94%. Additionally, we utilized the test cases from LogicAsker to design in-context learning demonstrations, which effectively enhance the logical reasoning capabilities of LLMs, e.g., improving from 75% to 85% for GPT-4.

By providing insights into the strengths and weaknesses of LLMs in reasoning, we are able to improve the reliability and trustworthiness of these models. The release of all the code and data aims to facilitate replication and encourage further research in this crucial area.

Limitations

This paper identifies two primary limitations that highlight areas for future research:

- Although our ICL (In-Context Learning) method significantly enhances the logical reasoning capabilities of large language models (LLMs), there remains a performance gap compared to human-level reasoning. Further refinements and innovations in model training and architecture may be necessary to bridge this gap.

- Our method is currently applicable only to LLMs that possess robust in-context learning capabilities. LLMs lacking this feature may not benefit from our approach. Future studies could explore fine-tuning methods to extend the applicability of our improvements across a broader spectrum of LLMs, potentially enhancing models with weaker or no inherent in-context learning abilities.

References

et al. Aarohi Srivastava. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *ArXiv*, abs/1905.13319.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *ArXiv*, abs/2207.04901.

Hugo Bronkhorst, Gerrit Roorda, Cor J. M. Suhre, and Martin J. Goedhart. 2020. Logical reasoning in formal and everyday reasoning tasks. *International Journal of Science and Mathematics Education*, 18:1673–1694.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314.

Catherine A. Gao, Frederick M. Howard, Nikolay S. Markov, Emma C. Dyer, Siddhi Ramesh, Yuan Luo, and Alexander T. Pearson. 2022. Comparing scientific abstracts generated by chatgpt to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers. *bioRxiv*.

Shuzheng Gao, Xinjie Wen, Cuiyun Gao, Wenxuan Wang, and Michael R. Lyu. 2023a. Constructing effective in-context demonstration for code intelligence tasks: An empirical study. *ArXiv*, abs/2304.07575.

Shuzheng Gao, Xinjie Wen, Cuiyun Gao, Wenxuan Wang, and Michael R. Lyu. 2023b. What makes good in-context demonstrations for code intelligence tasks with llms? *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 761–773.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan 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*, 9:346–361.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina 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. *ArXiv*, abs/2209.00840.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Xiaodong Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874.

Krystal Hu. 2023. Chatgpt sets record for fastest-growing user base. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/ Accessed: 2023-04-1.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *ArXiv*, abs/2212.10403.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *ArXiv*, abs/2310.01798.

Patrick J. Hurley and Lori Watson. 2020. A concise introduction to logic, 13/e.

Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *ArXiv*, abs/2301.08745.

Brenden M. Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Annual Meeting of the Association for Computational Linguistics*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yuexin Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *International Joint Conference on Artificial Intelligence*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. *ArXiv*, abs/2106.15772.

Linyong Nan, Chia-Hsuan Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryscinski, Nick Schoelkopf, Riley Kong, Xiangru Tang, Murori Mutuma, Benjamin Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir R. Radev. 2021. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Santiago Ontañón, Joshua Ainslie, Vaclav Cvicek, and Zachary Kenneth Fisher. 2022. Logicinference: A new dataset for teaching logical inference to seq2seq models. *ArXiv*, abs/2203.15099.

Barbara H. Partee, Alice ter Meulen, and Robert E. Wall. 1990. Mathematical methods in linguistics.

Arkil Patel, S. Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*.

Marco Tulio Ribeiro, Tongshuang Sherry Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *ACL*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *ArXiv*, abs/1608.01413.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *ArXiv*, abs/2210.01240.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Seyed Mehran Kazemi, Najoung Kim, and He He. 2023. Testing the general deductive reasoning capacity of large language models using ood examples. *ArXiv*, abs/2305.15269.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *ArXiv*, abs/1811.00937.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. In *Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *ArXiv*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *NeurIPS*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ioan Cristian Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685.

A Motivating Example

We provide a motivating example to illustrate the challenge of evaluating the logic reasoning ability of LLMs. Consider the following inference example: `Either it is raining, or Tom will play football; if it rains, then the floor will be wet; the floor is dry; therefore, Tom will play football.` We may encounter the following challenges: 1) If an LLM concludes correctly, it is unclear whether the response stems from reasoning or merely relies on simple heuristics such as memorization or word correlations (e.g., "dry floor" is more likely to correlate with "playing football"). 2) If an LLM fails to reason correctly, it is not clear which part of the reasoning process it failed (i.e., inferring not raining from floor being dry or inferring playing football from not raining). 3) There is a lack of a system that can organize such test cases to cover all other formal reasoning scenarios besides implication, such as logical equivalence (e.g., If A then B, if B then A; therefore, A if and only if B). 4) Furthermore, understanding an LLM's performance on such test cases provides little guidance on improving the reasoning ability of the LLM.

B Logical Rules and Fallacies

We list all the logic equivalence rules in Table 7-8, logic inference rules in Table 9, and common logical fallacies in Table 10.

C Extended Rules

C.1 Equivalent Extension

The equivalent rule extension is based on the following fact:

$$\{A \Leftrightarrow B, \forall x(A)\} \vdash \{\forall x(B)\}$$

(i.e., if A and B are equivalent, and for all x, A is true, then for all x, B is also true), and

$$\{A \Leftrightarrow B, \exists x(A)\} \vdash \{\exists x(B)\}$$

(i.e., if A and B are equivalent, and there exist x such that A is true, then there exist x such that B is true). For example, the predicate version of the DeMorgan's law

$$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$$

will become

$$\forall x(\neg(P(x) \wedge Q(x))) \Leftrightarrow \forall x(\neg P(x) \vee \neg Q(x)),$$

and

$$\exists x(\neg(P(x) \wedge Q(x))) \Leftrightarrow \exists x(\neg P(x) \vee \neg Q(x)).$$

In this example, the goal is to extend the propositional equivalence law to its predicate version by adding quantifiers. To achieve this goal, we first note that DeMorgan's law states that "P and Q cannot both be true" (e.g., Alice is happy and Bob is happy cannot both be true) is equivalent to "either not P or not Q" (e.g., either Alice is not happy or Bob is not happy). Since the two expressions are equivalent, we can add the same quantifier to both sides and the equivalence will still hold. Therefore, by adding a "for all" quantifier to both sides, we obtain "for all x, P(x) and Q(x) cannot both be true" (for all persons in the room, the person likes Charley and the person likes David cannot both be true) is equivalent to "for all x, either not P(x) or not Q(x)" (e.g., for all person in the room, either the person doesn't like Charley or the person doesn't like David). Before the extension, the law can only be applied to simple propositions (e.g., P = "Alice is happy", Q = "Bob is happy"), but after extension, the law can be applied to predicates with variables and quantifiers (e.g., P(x) = "x likes Charley", Q(x) = "x likes David") The same also applies to the "exist" quantifier.

C.2 Inference Extension

The inference rule extension is based on the following fact:

$$\{A \wedge B \rightarrow C\} \vdash \{\forall x, (A) \wedge \forall x, (B) \rightarrow \forall x, (C)\},$$

(i.e., if A and B imply C, then for all x, A is true and for all x, B is true implies for all x, C is true)

$$\{A \wedge B \rightarrow C\} \vdash \{\exists x, (A) \wedge \forall x, (B) \rightarrow \exists x, (C)\}.$$

(i.e., if A and B imply C, there exists x such that A is true and for all x, B is true implies there exists x such that C is true). Since all propositional inference rules are of the form $P \wedge Q \rightarrow C$, we can transform them into their predicate form $\forall x, P(x) \wedge \forall x, Q(x) \rightarrow \forall x, C(x)$ and $\exists x, P(x) \wedge \forall x, Q(x) \rightarrow \exists x, C(x)$ following similar procedure in the previous section.

D Natural Language Translation

D.1 Algorithm

Given an input: a logic clause of the form `[operator, Clause`$_A$`, Clause`$_B$`]`, where the

Table 7: Propositional logic equivalence laws.

Law	**Logical Equivalence**	**Example**
Idempotent laws	$P \wedge P \Leftrightarrow P$	I am a teacher and I am a teacher $\Leftrightarrow$ I am a teacher.
	$P \vee P \Leftrightarrow P$	It's raining or it's raining $\Leftrightarrow$ it's raining.
Commutative laws	$P \wedge Q \Leftrightarrow Q \wedge P$	It is cold and it is winter $\Leftrightarrow$ It is winter and it is cold.
	$P \vee Q \Leftrightarrow Q \vee P$	You can go to the party or you can study $\Leftrightarrow$ You can study or you can go to the party.
Associative laws	$(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$	It is raining and it is cold, and also it is winter $\Leftrightarrow$ It is raining, and also, it is cold and it is winter.
	$(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$	Either I will go to the park or I will go to the library is true, or I will go to the cinema $\Leftrightarrow$ I will go to the park or either I will go to the library or I will go to the cinema is true.
Distributive laws	$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$	It is raining and either I have an umbrella or I have a raincoat $\Leftrightarrow$ It is raining and I have an umbrella, or it is raining and I have a raincoat.
	$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$	Either I will go to the park, or it is cloudy and it is cold $\Leftrightarrow$ Either I will go to the park or it is cloudy is true, and either I will go to the park or it is cold is true.
DeMorgan's laws	$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$	It is not true that it's both cold and raining $\Leftrightarrow$ It's not cold or it's not raining.
	$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$	It's not true that I will study or play $\Leftrightarrow$ I won't study and I won't play.
Complement laws	$\neg(\neg P) \Leftrightarrow P$	It is not the case that it is not raining $\Leftrightarrow$ It is raining.
	$P \wedge \neg P \Leftrightarrow False$	It is raining and it is not raining.
	$P \vee \neg P \Leftrightarrow True$	It is raining or it is not raining.
Conditional laws	$P \rightarrow Q \Leftrightarrow \neg P \vee Q$	If it rains, then I'll stay at home $\Leftrightarrow$ It doesn't rain or I stay at home.
Bidirectional laws	$(P \leftrightarrow Q) \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$	I'll go to the park if and only if it's sunny $\Leftrightarrow$ Either it's sunny and I go to the park, or it's not sunny and I don't go to the park.
Identity laws	$P \wedge True \Leftrightarrow P$	It is raining and it is true $\Leftrightarrow$ It is raining.
	$P \vee False \Leftrightarrow P$	I will study or it's false $\Leftrightarrow$ I will study.

Table 8: Predicate logic quantifier laws.

Law	Logical Equivalence	Example
Quantifier Negation	$\neg\forall x P(x) \Leftrightarrow \exists x \neg P(x)$	It is not the case that all birds can fly $\Leftrightarrow$ There exists a bird that cannot fly.
	$\neg\exists x P(x) \Leftrightarrow \forall x \neg P(x)$	There is no human that can live forever $\Leftrightarrow$ All humans cannot live forever.
Quantifier Distribution	$\forall x(P(x) \land Q(x)) \Leftrightarrow \forall x P(x) \land \forall x Q(x)$	Every student is smart and diligent $\Leftrightarrow$ Every student is smart, and every student is diligent.
	$\exists x(P(x) \lor Q(x)) \Leftrightarrow \exists x P(x) \lor \exists x Q(x)$	There is a person who is either a doctor or a lawyer $\Leftrightarrow$ There is a person who is a doctor, or there is a person who is a lawyer.
Quantifier Commutation	$\exists x \exists y P(x, y) \Leftrightarrow \exists y \exists x P(x, y)$	There exists a child and a toy such that the child owns the toy $\Leftrightarrow$ There exists a toy and a child such that the child owns the toy.
	$\forall x \forall y P(x, y) \Leftrightarrow \forall y \forall x P(x, y)$	For all parents and children, the parent loves the child $\Leftrightarrow$ For all children and parents, the parent loves the child.
Quantifier Transposition	$\exists x \forall y P(x, y) \Leftrightarrow \forall y \exists x P(x, y)$	There exists a food that all people like is not generally equivalent to For all people, there exists a food that they like.
	$\forall x \exists y P(x, y) \Leftrightarrow \exists y \forall x P(x, y)$	For every person, there exists a food that they like is not generally equivalent to There exists a food that every person likes.
Quantifier Movement	$\forall x(P \to Q(x)) \Leftrightarrow (P \to \forall x Q(x))$	For every child, if it is raining then they are inside $\Leftrightarrow$ If it is raining, then every child is inside when the notion of raining doesn't depend on the specific child.
	$\exists x(P \land Q(x)) \Leftrightarrow (P \land \exists x Q(x))$	There exists a student who is tall and a good basketball player $\Leftrightarrow$ There is a tall student and there exists a student who is a good basketball player when the notion of being tall doesn't depend on the specific student.

Table 9: Propositional and predicate logic inference rules.

Inference Rule	Logical Form	Example
Universal Instantiation	$\forall x P(x) \vdash P(c)$	All birds have wings. Hence, this crow has wings.
Existential Generalization	$P(c) \vdash \exists x P(x)$	This apple is red. Hence, there exists a red apple.
Modus Ponens	$\{P \to Q, P\} \vdash Q$	If it rains, the street gets wet. It is raining. Hence, the street is wet.
Modus Tollens	$\{P \to Q, \neg Q\} \vdash \neg P$	If I study, I will pass the test. I did not pass the test. Hence, I did not study.
Transitivity	$\{P \to Q, Q \to R\} \vdash P \to R$	If it rains, I take my umbrella. If I take my umbrella, I won't get wet. Hence, if it rains, I won't get wet.
Disjunctive Syllogism	$\{P \lor Q, \neg P\} \vdash Q$	Either it's raining or it's snowing. It's not raining. Hence, it's snowing.
	$\{P \lor Q, \neg Q\} \vdash P$	Either it's raining or it's snowing. It's not snowing. Hence, it's raining.
Addition	$\{P\} \vdash P \lor Q$	It is raining. Hence, it is raining or it is snowing.
	$\{Q\} \vdash P \lor Q$	It is snowing. Hence, it is raining or it is snowing.
Simplification	$\{P \land Q\} \vdash P$	It is raining and it is cold. Hence, it is raining.
	$\{P \land Q\} \vdash Q$	It is raining and it is cold. Hence, it is cold.
Conjunction	$\{P, Q\} \vdash P \land Q$	It is raining. It is cold. Hence, it is raining and it is cold.
Constructive Dilemma	$\{P \to Q, R \to S, P \lor R\} \vdash Q \lor S$	If it rains, I'll stay at home. If I work, I'll be tired. Either it will rain or I'll work. Hence, I'll either stay at home or be tired.

Table 10: Common fallacies.

Name	Logical Form	Example
Affirming the Consequent	$p \to q, q \vdash p$	If I study, I will pass the test. I passed the test. Therefore, I studied.
Denying the Antecedent	$p \to q, \neg p \vdash \neg q$	If it rains, the street gets wet. It is not raining. Therefore, the street is not wet.
Affirming a Disjunct	$p \lor q, p \vdash \neg q$	Either I will study or I will fail the test. I studied. Therefore, I will not fail the test.
Denying a Conjunct	$\neg(p \land q), \neg p \vdash q$	I'm not both hungry and thirsty. I'm not hungry. Therefore, I'm thirsty.
Illicit Commutativity	$p \to q \vdash q \to p$	If I am in Paris, then I am in France. Therefore, if I am in France, I am in Paris.
Undistributed Middle	$\forall x(P(x) \to Q(x)), Q(a) \vdash P(a)$	All dogs are animals. My cat is an animal. Therefore, my cat is a dog.

clauses are also of the form [operator, Clause$_A$, Clause$_B$], the algorithm will do the following:

1. **Single Proposition Clause:** If the clause is just a single proposition, the algorithm finds this proposition's natural language form and returns it. The natural language form is obtained by combining vocabularies according to certain templates (e.g., subject + action).

2. **Negation:** If the clause starts with a "¬" operator, the algorithm then translates the rest of the clause based on a negation template, making sure to negate the statement.

3. **Quantifiers:** For clauses that start with "∀" (meaning for all items) or "∃" (meaning there is at least one item), it translates these into natural language, adjusting the phrasing based on whether we're asserting something positively or negating it.

4. **Logical Connectives:** If the clause combines propositions using logical operators like "∧", "∨", "→" (implies), or "↔" (if and only if), the function translates these into natural language phrases that express the relationship between the propositions.

D.2 Example

Consider the expression: [$\forall x$, $\rightarrow$, A(x), B(x)]. Here's how the function would translate it:

1. It sees the "$\forall x$" quantifier and adds "For all x," to the sentence and continues to process the clause [$\rightarrow$, A(x), B(x)].

2. It sees the "$\rightarrow$" operator, which means "if...then...". It connects the two operands with the operator and obtains "For all x, if A(x), then B(x)". Then, it continues to process the clauses A(x), B(x).

3. Since A(x), B(x) are single proposition clauses, the function looks up the vocabulary and synthesizes the natural language versions of the proposition. For example, A(x) = "x drinks water", B(x) = "x is a cashier".

4. It constructs the sentence: "For all x, if x drinks water, then x is a cashier".

D.3 Vocabulary

We list the vocabulary used in our experiment:

Subjects

- x, y, z, James, Mary, Robert, Patricia, John, Jennifer, Michael, Linda, William, Elisabeth, David, Barbara, Richard, Susan, Joseph, Jessica, Thomas, Sarah, Charles, Karen, Alice, Benjamin, Daniel, Emily, George, Helen, Ian, Julie.

Predicates

- a cashier, a janitor, a bartender, a server, an office clerk, a mechanic, a carpenter, an electrician, a nurse, a doctor, a police officer, a taxi driver, a soldier, a politician, a lawyer, a scientist, an astronaut, a poet, an artist, a sailor, a writer, a musician, poor, rich, happy, sad, fast, curious, excited, bored, tired, joyful, intelligent, skilled, efficient, meticulous, creative.

Actions

- make tea, makes tea, making tea, drink water, drinks water, drinking water, read a book, reads a book, reading a book, play tennis, plays tennis, playing tennis, play squash, plays squash, playing squash, play a game, plays a game, playing a game, go running, goes running, running, work, works, working, sleep, sleeps, sleeping, cook, cooks, cooking, listen to a song, listens to a song, listening to a song, write a letter, writes a letter, writing a letter, drive a car, drives a car, driving a car, climb a mountain, climbs a mountain, climbing a mountain, take a plane, takes a plane, taking a plane, paint a picture, paints a picture, painting a picture.

Impersonal Candidates

- snowing, snows, doesn't snow, snow, raining, rains, doesn't rain, rain, sunny, is sunny, is not sunny, be sunny, cloudy, is cloudy, is not cloudy, be cloudy, windy, is windy, is not windy, be windy, cold, is cold, is not cold, be cold, late, is late, is not late, be late, overcast, is overcast, is not overcast, be overcast, foggy, is foggy, is not foggy, be foggy, humid, is humid, is not humid, be humid.

E Demonstration Examples

The following is a three-shot demonstration example used in our experiment:

Consider the following premises: For all u, u is not a doctor. There is at least one u for which the claim that u is a doctor and the claim that u will drive a car cannot both be true. We cannot infer that: There is at least one u for which u will drive a car. Because this is a logical fallacy of the existential denying a conjunct.

Consider the following premises: For all x, x will not drive a car or x will play tennis. There is at least one x for which x will drive a car or x will drink water. We cannot infer that: There is at least one x for which x will play tennis. Because the conclusion is not related to the premises

Consider the following premises: For all x, x is a scientist or x is curious. There is at least one x for which x is not a scientist or x will sleep. We cannot infer that: There is at least one x for which x is curious. Because the conclusion is not related to the premises

Now answer the following question:

Consider the following premises: For all x, x will work or x is a poet. For all x, x will not work. Can we infer the following from them? Answer yes or no: There is at least one x for which x is not a poet.

F Accessing LLMs

To access these LLMs, we use the OpenAI APIs of GPT-4 [2] (`gpt4`), ChatGPT [3] (`gpt-3.5-turbo`) and GPT-3[4] (`text-davinci-003`), the webpage of Bard [5], and the open-source weights of Vicuna-13b [6] and Guanaco-33b [7]. For GPT families, we use default hyper-parameters in the APIs.

G Prompting LLMs

We prompt the LLMs to answer the test cases generated by LogicAsker. The prompt template we used is "Consider the following premises: [Premises]. Can we infer the following from them? Answer yes or no: [Conclusion]". We set the system prompt of GPT APIs to blank.

[2] https://openai.com/gpt-4
[3] https://openai.com/blog/chatgpt/
[4] https://beta.openai.com/docs/models/gpt-3
[5] https://bard.google.com/
[6] https://lmsys.org/blog/2023-03-30-vicuna/
[7] https://huggingface.co/timdettmers/guanaco-33b-merged