A Closer Look at Tool-based Logical Reasoning with LLMs: The Choice of Tool Matters

Long Hei Matthew Lam **Ramya Keerthy Thatikonda** **Ehsan Shareghi**
DSAI, Monash University
llam0013@student.monash.edu.au Ramya.Thatikonda1@monash.edu
Ehsan.Shareghi@monash.edu

Abstract

The emergence of Large Language Models (LLMs) has demonstrated promising progress in solving logical reasoning tasks effectively. Several recent approaches have proposed to change the role of the LLM from the reasoner into a translator between natural language statements and symbolic representations which are then sent to external symbolic solvers to resolve. This paradigm has established the current state-of-the-art result in logical reasoning (i.e., deductive reasoning). However, it remains unclear whether the variance in performance of these approaches stems from the methodologies employed or the specific symbolic solvers utilized. There is a lack of consistent comparison between symbolic solvers and how they influence the overall reported performance. This is important, as each symbolic solver also has its own input symbolic language, presenting varying degrees of challenge in the translation process. To address this gap, we perform experiments on 3 deductive reasoning benchmarks with LLMs augmented with widely used symbolic solvers: Z3, Pyke, and Prover9. The tool-executable rates of symbolic translation generated by different LLMs exhibit a near 50% performance variation. This highlights a significant difference in performance rooted in very basic choices of tools. The almost linear correlation between the executable rate of translations and the accuracy of the outcomes from Prover9 highlight a strong alignment between LLMs ability to translate into Prover9 symbolic language, and the correctness of those translations. [1]

1 Introduction

The recent state-of-the-art approaches to logical reasoning have combined Large Language Models (LLMs) with external symbolic mechanisms (Nye et al., 2021; Pan et al., 2023; Ye et al., 2023; Gao et al., 2023; Lyu et al., 2023). This approach leverages LLMs' remarkable proficiency in translating natural language into symbolic representation such as First Order Logic (FOL) or symbolic solvers' specified language (e.g., Pyke, Z3) (Yang et al., 2023), and the symbolic solver's ability to execute these translations through a fully deterministic proof process (Metaxiotis et al., 2002). These existing published methods try a variety of tools and tool-specific formalism. Table 1 summarises various tools used in recent state-of-the-art studies. This variability of tools makes it impossible to have a fair understanding of each approach. There is currently a lack of consistent comparison that will allow others to understand better where this performance gain stems from.

In this paper, we take 3 widely used tools: Z3 (de Moura and Bjørner, 2008), Pyke (Frederiksen, 2008), and Prover9 (McCune, 2005) and analyse the difficulty LLMs face for translating natural language into their desired input format, and the internal capability of these tools at solving certain satisfiability tasks. We select GPT4o, GPT-3.5-Turbo (OpenAI, 2023), Gemini-1.0-Pro (Team et al., 2023) and Cohere Command R Plus, as representatives of the most capable family of LLMs, along with 3 widely used deductive reasoning benchmarks ProofWriter (Tafjord et al., 2021), FOLIO (Han et al., 2022), and ProntoQA (Saparov and He, 2023). We conduct a fair side-by-side comparison of tools by trying various number of identical prompts, demonstration shots, and minimal adjustment for each solver.

Our findings indicate that LLMs find it easier to translate for Prover9, followed by Z3, and lastly Pyke. Although Prover9 can solve more questions accurately, Prover9 demonstrates a lower discrepancy between execution rate and overall accuracy. This means that Prover9 is more likely to solve a question given the right syntax and format produced by LLMs. Overall, Z3 and Prover9 are all

[1] Code and data are publicly available at https://github.com/Mattylam/Logic_Symbolic_Solvers_Experiment.

Proceedings of The 22nd Annual Workshop of the Australasian Language Technology Association, pages 41–63

Solver	Dataset	Papers	Problem
Z3	AR-LSAT (Zhong et al., 2022), ProntoQA (Saparov and He, 2023), ProofWriter (Tafjord et al., 2021), BoardgameQA (Kazemi et al., 2023)	LogicLM, SatLM	Analytical, Deductive, FOL
Pyke	ProntoQA (Saparov and He, 2023), ProofWriter (Tafjord et al., 2021)	LogicLM, Logical Solver	Deductive, FOL
Prover9	FOLIO (Han et al., 2022)	LogicLM, LINC	Deductive, FOL

Table 1: A summary of the symbolic solvers and the datasets it has solved in different studies: LogicLM (Pan et al., 2023), LINC (Olausson et al., 2023), Logical Solver (Feng et al., 2023), and SatLM (Ye et al., 2023).

competitive options, Pyke's performance is significantly inferior and only comparable to the other tools in solving PrOntoQA. Our experiments across 3 benchmarks (based on the accuracy of outputs) highlight an up-to 50% of performance variation for each LLM under different tools, and well as the performance change for each tool under different LLMs.

2 Tools & Logical Reasoning with LLMs

The tool-based approaches to logical reasoning combine LLMs with external symbolic solvers. This synergy harnesses the capability of LLMs to convert diverse natural language statements into logical symbolic formalism. While being less flexible compared with free-form reasoning methods, such as Chain-of-Thought (Wei et al., 2022), the tool-based approach, given a *correct formal translation*, has important advantages: logical coherence during the reasoning (i.e., unlike LLMs, theorem provers cannot make reasoning shortcuts or hallucinate) is guaranteed, while the internal proof trace of the theorem provers offers a transparent and verifiable reasoning chain.

2.1 Logical Solvers

Automated theorem provers (ATPs) and Satisfiability Modulo Theories (SMT) solvers are tools equipped with built-in functions designed to assist in logical reasoning tasks. These solvers can vary in syntax, proof search strategies, theorem automation, and complexity. ATPs efficiently resolve first order logic problems without external interaction. SMT solvers closely resemble ATPs in solving first-order formulae but add complexity by handling theories such as equality, arrays, and bit-vectors. Logical solvers, specifically Z3, Prover9, and Pyke, are used for logical reasoning tasks with LLMs due to their ease of use in a Python environment (Pan et al., 2023; Ye et al., 2023). We study the logical solvers based on their ability to handle first-order logic and explore the crucial differences in external syntax and internal theories of these tools. In this context, we define the task as follows: given a set of premises $P \in \{P_1, P_2, \ldots, P_n\}$, the objective is to determine whether the conclusion C logically follows from these premises. The translation syntax for each tool is presented in Figure1.

Z3 Prover developed by Microsoft, is an SMT solver designed to determine the satisfiability of given constraints (de Moura and Bjørner, 2008). Z3 encompasses a diverse array of functionalities, including equality reasoning, arithmetic operations, handling arrays, and incorporating quantifiers. It supports multiple programming languages and mathematical operators, making it a versatile tool for a wide range of research applications. Z3 utlizes the DPLL algorithm for satisfiability resolution, where constrains are converted to conjunctive normal form (CNF). The solver then searches for a solution through backtracking, continuing until it finds a combination of truth values that satisfies the conditions. In deductive logical reasoning, the tool can check if the conclusion C renders the assertions P satisfiable. Z3 requires an explicit specification of data types of variables, functions, and their attributes, which are typically Boolean for deductive reasoning. Due to its flexible operations, Z3 has been applied to tasks beyond logical verification, as shown in Table 1. Additionally, the simplicity of these tasks enables the translation format of Z3 to resemble programming languages, as demonstrated in Appendix A.2.

Prover9 is an automated theorem prover for first-order and equational logic, based on resolution techniques (McCune, 2005). This tool accepts first-order logic statements and applies logical transformations such as CNF conversion, quantifier operations, and skolemization to produce simplified clauses. The inference process involves iterating over given clauses to generate new clauses in a non-redundant manner by categorizing the clauses into usable and non-usable forms. For deductive reasoning task, the premises P produce new premise, i.e., $\{P_1, P_2\} \implies \{P_{12}\}$, for various combinations. These derived premises P_{xy} are retained if they are relevant to the conclusion, and discarded otherwise. The inference is based on all the stored premises once all combinations have been exhausted. Although the logical transformations allow flexible input, Prover9 is sensitive to special characters and

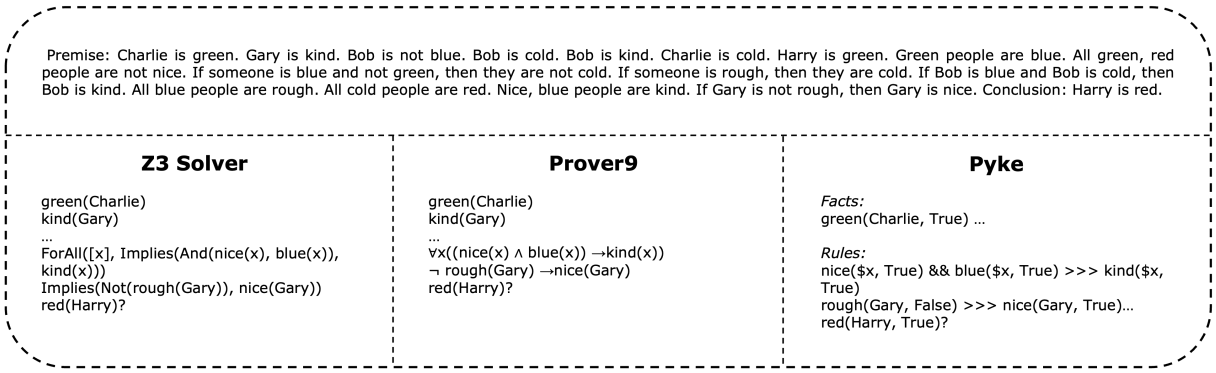

Figure 1: Overview of syntax used for different Theorem Provers: Z3 and Prover9 adhere to the traditional first-order logic (FOL) format, while Pyke adopts a simplified formula approach, distinguishing premises into rules and facts

spaces, which require careful handling. Compared to Z3 or other ATPs, Prover9 cannot solve a variety of mathematical problems, thus limiting its applicability to certain fields of logic (McCune, 2003). In Python, Prover9 is accessible through the NLTK logic library.

Pyke short for Python Knowledge Engine, is a solver used for building and executing rule-based expert systems (Frederiksen, 2008). Although pyke is used for optimizing software development, Pan et al. (2023) demonstrated its application in solving a first-order logic problem. Given a logical inference task, Pyke establishes a knowledge base and incorporates known facts (`fact.kfb`) and rules (`rule.krb`) from the input, i.e., $P \rightarrow (P_{\text{facts}}, P_{\text{rules}})$. The conclusion is parsed as a rule that is propagated through the knowledge base until it reaches a resolution. The predicates in the first order logic are treated as facts and are connected to form rules. Given its limited syntax, Pyke supports simple connectives such as 'and', 'or', and 'implies'. The free variables (e.g., $\$x$) are generally considered to be universal quantifiers, thus restricting the use of existential quantifiers. Due to these limitations, Pyke may not adequately handle complex tasks involving first-order logic, such as FOLIO. However, it remains well-suited for rule-based tasks like ProofWriter and ProntoQA.

2.2 Free-form Logical Reasoning with LLMs

The free-form approaches to reasoning rely on LLMs' internal capabilities via various mechanisms to help improve LLM's performance in logical reasoning. For example, prompts that encourage LLMs to solve tasks in a Chain-of-Thought approach is a general technique that enhances LLM's performance (Wei et al., 2022; Kojima et al., 2022). Despite the promising outcomes, this approach falls short when dealing with complex logical reasoning tasks. This limitation stems from the lack of explicit logical grounding and the inherent ambiguous and nuanced nature of natural language. Recent studies have revisited Formal Logic to address this challenge. Han et al. (2022) shows that incorporating first-order logic (FOL) translations into the context can notably enhance LLM's performance. Feng et al. (2023) emulates the reasoning processes of an automated theorem solver (Pyke) through solving Logical tasks using the tool-based approach and training LLMs on Pyke's reasoning steps. The free-form approach capitalises on the inherent capabilities of LLM to learn complex logical rules. However, this approach solely relies upon LLM's logical reasoning prowess and is susceptible to issues such as hallucinations and taking shortcuts (Dasgupta et al., 2022; Ji et al., 2023). To address this issue, recent approaches aim to augment LLMs with external symbolic solvers (Ye et al., 2023; Gao et al., 2023).

2.3 Tool-based Logical Reasoning with LLMs

Ye et al. (2023) and Gao et al. (2023) integrated Z3 and Python interpreters with LLMs to tackle various reasoning datasets. Pan et al. (2023) expanded upon this by incorporating a broader range of symbolic solvers and employing error-solving self-refinement techniques. However, the rationale behind the adoption of symbolic solvers primarily relied on theoretical definitions rather than empirical performance evaluations. Consequently, there exists a gap in the literature regarding the exploration of the interplay between LLMs, symbolic solvers, and their respective performance characteristics.

The primary advantages of the tool-based ap-

proach are: (1) The tasks are now processed with clear logical grounding and unambiguous language. This approach guarantees that the answer is not a product of hallucination or shortcuts, because the symbolic tools will exhaustively process all logical rules in the premise and only execute clear and correct commands. (2) As LLM's translation capability continues to improve, the tool-based approach will be able to solve more complex logical problems, provided they fall within the logical reasoning capacity of symbolic solvers. (3) The tool errors are clearly labeled and displayed (i.e., run-time error messages). This allows the introduction of various error-solving mechanisms like self-refinement (Pan et al., 2023). In contrast, it is difficult for the free-form approach to improve upon its current results in the absence of any reliable feedback, specially in the light of recent debates on LLMs self-correction capability (Huang et al., 2024; Li et al., 2024). In this study, errors are isolated into solver-specific errors (e.g., LLM's translation misses a bracket, which causes the solver to throw an error) and parse errors (i.e., Predicate extraction mistakes or LLMs interpreting the logical statements incorrectly, examples of these are shown in Appendix A.3).

The main disadvantages of the tool-based approach are: (1) This approach does not apply to tasks that do not have a complete reasoning chain. All symbolic solvers require a full chain of logic to reach the correct conclusion. For instance, consider the followint example: *Premise: People like Mark love bbq. Question: Mark is not Human?* Both humans and LLMs can answer this question correctly, but a tool-based approach will fail. This is due to the break in the chain of logic. The term "Mark is human" is missing from the premise. Although this term is obvious for humans and LLMs, symbolic solvers require the exact match in predicates to process the task. A detailed discussion of this issue is included in section 3.2. (2) Changes in LLMs can cause solver-specific errors.[2] (3) This approach is unforgiving to simple translation errors. While processing logical tasks, Human and LLMs can often bypass errors to some extent and still reach the correct conclusion. However, a tool-based approach requires the LLM to translate tasks flawlessly, even minor mistakes like misusing suffixes (e.g., "Jompuses(x)" instead of "Jompus(x)") will cause the symbolic solver to throw an error. One of the main focuses of this study is the analysis of how different symbolic tools handle errors caused by LLMs.

[2]For instance, during the experiment stage, we tried to rerun the SatLM experiment on ProofWriter, but the execution rate dropped from 99% to 20%. This is caused by GPT3.5 not being able to add a complete bracket to the method Forall()). It is a surprising mistake that continues to happen.

3 Experiments

3.1 Experimental Setup

In our experiments we assess the performance variations of LLM when paired with various symbolic solvers. We evaluate GPT-4o, GPT-3.5-Turbo, Gemini-Pro-1.0, and Command-r-plus integrated with Z3, Pyke, Prover9 on three common logical reasoning benchmarks (introduced shortly). Unlike Pan et al. (2023) and other studies, we exclude self-refinement methods and random guessing procedures. In cases where LLM's translation is infeasible, it will not yield an answer, and any specific errors encountered are documented. The only exception is the missing bracket issue for the translation of Z3, as this was not an issue in experiments done in Ye et al. (2023) and Pan et al. (2023). We use a one-shot demonstration for all experiments. If different solvers are employed to tackle the same dataset, the given prompt problem remains consistent, with the sole variance lying in the solver-specific translations of the prompts. Examples of the prompt are shown in Appendix A.2. We also expand the one-shot experiment for FOLIO to two-shot and four-shot to highlight the impact of additional shots. The primary metrics for evaluation consist of two key factors: the percentage of executable logical formulations (ExecR.), and the overall accuracy (Acc).

Data The 3 benchmarks are introduced shortly and examples are included in Appendix A.1. We limit the test set size to 200 for cost reason. **PrOntoQA** (Saparov and He, 2023) is a synthetic dataset created to analyze the capacity of LLMs for deductive reasoning. We use the hardest fictional characters version and the hardest 5-hop subset for evaluation. PrOntoQA only has questions in the close world setting (i.e., True/False only). We include this dataset in the experiment to compare natural and fictional settings, as it has a similar level of logical difficulty to ProofWriter. **ProofWriter** (Tafjord et al., 2021) is a commonly used dataset for deductive logical reasoning. Compared with PrOntoQA, the problems are expressed in a more naturalistic language form. We evaluate 6 different variations of ProofWriter. We use both open-world

Dataset	LLMs	Z3		Prover9		Pyke	
		ExecR.	Acc.	ExecR.	Acc.	ExecR.	Acc.
ProofWriter (Avg. OWA)	gpt-4o	75.00%	74.17%	97.33%	95.67%	99.83%	79.17%
	gpt-3.5-turbo	84.83%	82.88%	90.67%	87.00%	62.83%	53.33%
	gemini-1.0-pro	93.00%	91.00%	86.83%	62.50%	49.33%	36.67%
	command-r-plus	88.67%	87.00%	61.33%	56.66%	61.83%	51.50%
ProofWriter (Avg. CWA)	gpt-4o	77.83%	77.83%	98.00%	98.00%	99.83%	87.00%
	gpt-3.5-turbo	88.33%	88.00%	94.00%	93.83%	58.17%	51.67%
	gemini-1.0-pro	96.83%	96.83%	84.83%	58.50%	42.83%	34.17%
	command-r-plus	92.50%	92.50%	58.67%	58.33%	45.33%	41.33%
PrOntoQA	gpt-4o	96.00%	96.00%	100.00%	100.00%	100.00%	100.00%
	gpt-3.5-turbo	95.50%	93.49%	85.50%	63.50%	99.50%	72.50%
	gemini-1.0-pro	100.00%	100.00%	100.00%	97.50%	100.00%	100.00%
	command-r-plus	93.00%	87.00%	64.50%	46.50%	96.50%	92.00%
FOLIO	gpt-4o	40.00%	36.00%	84.00%	66.50%	✗	✗
	gpt-3.5-turbo	29.00%	24.49%	61.00%	39.99%	✗	✗
	gemini-1.0-pro	31.00%	25.50%	67.50%	50.00%	✗	✗
	command-r-plus	25.50%	19.00%	50.50%	32.50%	✗	✗
Combined	gpt-4o	74.31%	73.50%	94.06%	91.71%	99.86%	85.50%
	gpt-3.5-turbo	80.50%	78.83%	87.56%	80.75%	66.07%	55.36%
	gemini-1.0-pro	87.56%	86.12%	85.31%	63.81%	53.79%	44.64%
	command-r-plus	82.75%	80.56%	59.38%	53.00%	60.64%	52.93%

Table 2: Accuracy and execution rate of 1-shot experiments done with gpt-4o, gpt-3.5-turbo, gemini-pro-1.0 and command-r-plus on 3 Datasets. Results for Proofwriter Open and Closed World Assumptions (OWA and CWA) are averaged over depths (Depth 2, 3, and 5). We present the percentage of executable logical formulations (ExecR.) together with the overall accuracy (Acc.). ✗: the tool was unable to solve this dataset. The numbers highlighted in red color represent the highest accuracy between the 3 chosen tools.

(OWA) and close-world assumptions (CWA), including depth-2, depth-3, and depth-5 (i.e., each part requiring 2, 3, and 5 hops of reasoning). To ensure a fair evaluation, we control all datasets to have a uniform distribution of True, False, and Unknown (if applicable) answers. **FOLIO** (Han et al., 2022) is a difficult expert-written dataset for first-order logical reasoning. The problems are mostly aligned with real-world knowledge and expressed in natural flowing language. Tackling its questions demands adeptness in complex first-order logic reasoning. Pyke is unable to solve FOLIO, this is due to the lack of a built-in function for the exclusive disjunction (i.e., either-or). In contrast, Prover9 and Z3 offer a built-in function to handle this logic seamlessly.

3.2 Main Results

We report the results of the tool-based reasoning approach experiments in Table 2. Different LLMs exhibit varying preferences for tools. For datasets with simpler logical complexity, GPT models tend to favor Prover9, while Gemini and Command R+ models perform significantly better using Z3. Pyke is only competitive in solving PrOntoQA and is unable to solve datasets like FOLIO and performs significantly worse on ProofWriter. Pyke's primary issue is the low and inconsistent executable rate. According to Table 4, without considering the option of LLMs, Prover9 performs better for the FOLIO dataset, Z3 performs better on other datasets. Both Z3 and Prover9 have their distinct advantages. Prover9's programming language, which closely resembles the language of First-Order Logic (FOL), contributes to its higher execution rate. The Pearson correlation coefficient between executable rate and accuracy across all LLMs (Prover9, Z3, and Pyke have, 0.98, 0.82, 0.94, respectively[3]) indicate an almost linear dependence between execution success and accuracy for Prover9. The lower cor-

[3]p-values: 1.02×10^{-18}, 5.37×10^{-7}, 6.1×10^{-12}

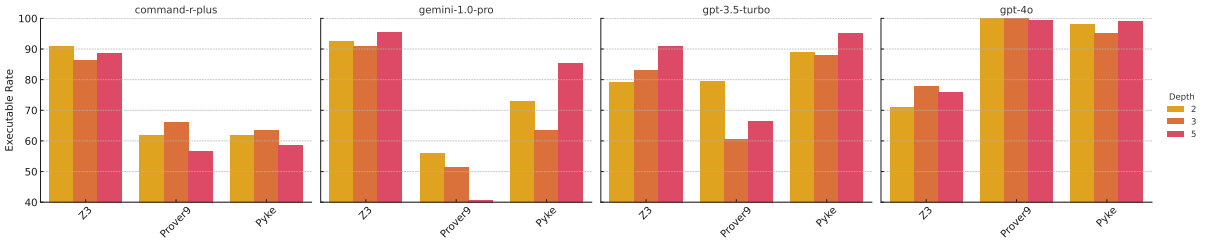

Figure 2: Executable Rate for different LLM-Tool combinations, for depth 2, 3, 5 of the ProofWriter Open World Assumption (OWA). Similar trend exists for the Close World Assumption (CWA).

relation for Z3 highlights the discrepancy between writing executable codes for Z3, and the accuracy of those codes.

Natural vs. Fictional We compare the performance of ProntoQA Depth 5 and ProofWriter CWA Depth 5 to investigate how different symbolic solvers affect the performance of tool-augmented LLMs in natural versus fictional world settings. The main difference between the datasets is that PrOntoQA uses fictional characters (i.e., imaginary characters like Jompus and Wompus), while ProofWriter is expressed in more naturalistic language. Saparov et al. (2023) have shown that real-world knowledge helps LLMs in reasoning more effectively, a fictional world setting decreases LLM's logical performance. On average, Prover9's performance is most aligned with this observation. The executable rate on average decreases for all LLMs, and average accuracy drops by 1.38% in a fictional setting. Both Z3 and Pyke's overall accuracy increased by 6.62% and 30.87%. This shows that while using Z3 and Prover9, fictional wording helps LLMs in generating consistent and correct translations. Overall, in a fictional setting, Pyke's performance is significantly boosted. Meanwhile, GPT-3.5-Turbo shifts its preference from Prover9 to Z3, and Command R+ changes its preference to Pyke. We speculate the nuance in results to be reflective of potential interference between commonsense knowledge and fictional statements.

Depth The relaiton between depth and executable rate is somewhat mixed, specially between depth 2 and 3. While for command-r-plus we observe a general decay in performance (i.e., between depth 2 and 5) across all tools, both GPT models and Gemini exhibit resilence to depth, with performance even improving across most tools (except for Prover9). This observation highlights the robustness of translation-based approaches (i.e., using LLMs for translation and tools for solving) in handling various complexities, while prior findings reported the reasoning ability of LLMs (alone) generally diminish as the number of reasoning hops increases (Han et al., 2022).

Demonstration Shots We present the statistics of the FOLIO dataset in varying number of shots in Table 3. Prover9 achieves the best performance, while Z3 struggles with execution rate. The best result for FOLIO was 66.5%, which is achieved with 1 shot prompting using GPT-4o and Prover9. The primary factors that limit the execution rate performance on FOLIO are: (1) some natural wordings in FOLIO make it difficult for predicate extraction. For example, GPT4o interpreted the term "Eastern wild turkey" as two separate terms "Eastern(x)" and "WildTurkey(x)", but "Eastern(x)" has no meaning and the predicate should be extracted as EasternWildTurkey(x). (2) FOLIO is annotated by humans and thus assumes a degree of commonsense, this presents incomplete reasoning chains and ambiguous sentences. As shown in A.3, GPT-3.5-Turbo incorrectly translated the statement "Marvin cannot be from Earth and from Mars." into "Not(And(FromEarth(marvin), FromMars(marvin)))", which entails Marvin is not from Earth and not from Mars. The simple fix is just to change Not() into Xor(). This problem was caused by the inherently ambiguous nature of the natural language. (3) there is a limitation to learning by increasing the number of shots. Specifically, GPT-4o and Prover9's parse errors increased with a higher number of shots, as shown in Table 3. Overall, while Prover9 can solve a greater number of questions, Z3 shows significant potential in addressing FOLIO. This is due to Z3's error-display capabilities, which are essential for continuous improvement.

	Z3		Prover9	
	ExecR.	Acc.	ExecR.	Acc.
GPT-4o				
$k = 1$	40%	36%	84%	66.5%
$k = 2$	50.5%	40.96%	74.5%	58%
$k = 4$	51%	39.5%	77%	62%
GPT-3.5-Turbo				
$k = 1$	29%	24.49%	61%	39.99%
$k = 2$	37%	31%	58%	40.5%
$k = 4$	48%	36.5%	65%	44.5%
Gemini-1.0-Pro				
$k = 1$	31%	25.5%	67.5%	50.00%
$k = 2$	47.5%	39%	60.5%	38%
$k = 4$	48%	36.5%	65.5%	44%
Command-R-Plus				
$k = 1$	25.50%	19.00%	50.50%	32.50%
$k = 2$	33.5%	26.5%	42.5%	29.5%
$k = 4$	42%	32.5%	60.5%	46%

Table 3: The effect of varying number of shots ($k = 1, 2, 4$) on accuracy and executable rates under GPT-4o, GPT-3.5-turbo, Gemini-1.0-pro and command-r-plus on FOLIO. We present the percentage of executable logical formulations (ExecR.) together with the overall accuracy (Acc.).

4 Analysis

As indicated by the executable rate in Table 2, LLMs generally find it easier to produce executable logical formulations for Prover9. This is attributed to its foundation in FOL-based programming language, which most large language models (LLMs) are familiar with as a form of logical formulation. While GPT models are more successful at converting these logical formulations into accurate results, Gemini-1.0-pro and Command R+ face challenges in achieving similar accuracy. This is an issue because an executable formulation cannot provide feedback when an incorrect result is given. This hinders further improvement and self-refinement. Z3 does not have this issue. Its executable rate is a reflection of its accuracy. Moreover, Z3's programming language closely aligns with Python, offering a unique advantage in error displaying and further improvement. Z3 is also a flexible tool that allows the inclusion of self-defined complex logical rules like "XorAnd()" (i.e., a combination of the rule "Either or" and "And".). This capability is particularly useful for addressing complex reasoning datasets like FOLIO. We did not define such a rule during our experiment but this capability should be considered in further studies.

Non-executable logical formulations can be categorized into *parse errors* and *execution errors*. Additionally, for Z3, there is a separate category known as *execution exceptions*.

- ***parse error*** refers to the mistakes identified by the parser. Through the prompt, we have predefined a set of instructions and logical rules that LLMs can use. However, when LLMs hallucinate and generate logical rules or code that do not exist in the solver, the parser will detect these discrepancies and throw an parse error. This error indicates the LLM's inability to adhere to the one-shot prompt, resulting in methods or code that the parser cannot process. For instance, using Exist() instead of Exists() for Z3 is an example of such an error.
- ***execution error*** occurs when the solver encounters given facts that are inconsistent, predicates that are defined wrong, or when there are solver-specific syntax errors. This type of error can be resolved through self-refinement, as the errors are explicitly displayed. We call this run-time error.
- ***execution exception*** is a special case for Z3, where the solver runs both the original conclusion and the negation of the same conclusion but receives true as the answer in both cases. This indicates that the facts are inconsistent. We combined these errors into run-time errors for Figure 3 Z3 visualisation.

As shown in Figure 3, for GPT4o, while Pyke produced 3 execution errors on easier logical reasoning datasets in total, its high execution rate did not translate to high accuracy. Predominately Prover9 and Z3's error is a parse error, with execution error controlled at around 8 questions. In addition, all non-executable questions are different, there are no common questions that all 3 solvers find difficult to solve. For FOLIO, the execution error increases, and the parse error drops significantly. Challenging datasets, such as FOLIO, encompass a larger number of unseen, complex logical rules and more intricate predicates, which result in higher error rates during translation by LLMs. Additionally, there is an increasing number of questions that both solvers are unable to process. This suggests that

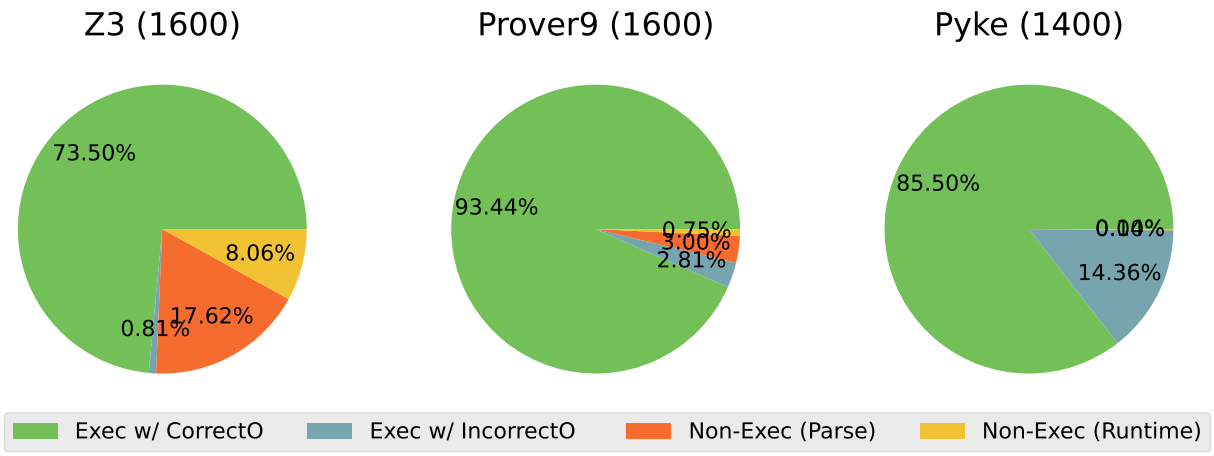

Figure 3: The proportion of various executable and non-executable instances per each tool for GPT4o. Note, Pyke does not include FOLIO (hence 1400 instances compared to Z3 and Prover 9). The *Exec w/ CorrectO*, and *Exec w/ IncorrectO* denote Executable translations that lead to correct, and incorrect outputs once executed by the tool. The *Non-exec (Parse) or (Runtime)* denote the non-executable translations which are either due to parsing error or other potential runtime issues.

both solvers find around 25-30% of questions hard to solve.

5 Conclusion

In this study, we investigated and compared the performance of LLMs combined with three widely used symbolic solvers to closely examine how each solver influences the performance of tool-augmented LLMs in logical reasoning. Our experiments demonstrated that the choice of tools (i.e., Z3, Pyke, Prover9) has a significant impact on the downstream performance across various benchmarks and LLMs.

Limitations

The tool-based approach to logical reasoning is limited to deductive reasoning datasets with a complete reasoning chain. This constraint arises from the inherent nature of symbolic solvers. A potential solution is for LLMs to generate the missing segments of the reasoning chain. Additionally, black-box LLMs can exhibit inconsistencies, producing results that change in the course of time. For instance, during our experiment, GPT-3.5-Turbo consistently failed to add a closing bracket to the method "Forall()", while Command R+ failed to include an opening bracket. This was not an issue for Pan et al. (2023) and Ye et al. (2023) (or at least was not reported in their papers). We limited our use of solvers to their built-in functions. To enhance the performance of each tool, more unique logical combinations can be integrated and implemented. For example, Z3 is a flexible tool that allows the inclusion of rules such as "Male(x) == Not(Female(x))". There is further potential to include more defined complex logical rules that can make LLM translation easier.

References

Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. 2022. Language models show human-like content effects on reasoning. *CoRR*, abs/2207.07051.

Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer.

Jiazhan Feng, Ruochen Xu, Junheng Hao, Hiteshi Sharma, Yelong Shen, Dongyan Zhao, and Weizhu Chen. 2023. Language models can be logical solvers. *arXiv preprint arXiv:2311.06158*.

Bruce Frederiksen. 2008. Applying expert system technology to code reuse with pyke. *PyCon: Chicago*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu,*

Hawaii, USA, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

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 Radev. 2022. FOLIO: natural language reasoning with first-order logic. *CoRR*, abs/2209.00840.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Loka Li, Guangyi Chen, Yusheng Su, Zhenhao Chen, Yixuan Zhang, Eric P. Xing, and Kun Zhang. 2024. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *CoRR*, abs/2402.12563.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *CoRR*, abs/2301.13379.

William McCune. 2003. Otter 3.3 reference manual. *arXiv preprint cs/0310056*.

William McCune. 2005. Release of prover9. In *Mile high conference on quasigroups, loops and nonassociative systems, Denver, Colorado*.

Kostas S. Metaxiotis, Dimitris Askounis, and John E. Psarras. 2002. Expert systems in production planning and scheduling: A state-of-the-art survey. *J. Intell. Manuf.*, 13(4):253–260.

Maxwell I. Nye, Michael Henry Tessler, Joshua B. Tenenbaum, and Brenden M. Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25192–25204.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3806–3824. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. 2023. Testing the general deductive reasoning capacity of large language models using OOD examples. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023. Harnessing the power of large language models for natural language to first-order logic translation. *CoRR*, abs/2305.15541.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2306–2319.

A Appendix

A.1 Dataset Examples

ProofWriter
Example: ProofWriter Depth 5 Open World Assumption Q774

Problem:
The bald eagle is blue. The bald eagle is kind. The bald eagle likes the cat. The bald eagle does not visit the tiger. The cat chases the mouse. The cat is green. The cat likes the bald eagle. The cat likes the mouse. The cat does not like the tiger. The mouse likes the cat. The tiger chases the cat. The tiger chases the mouse. The tiger is red. The tiger likes the cat. The tiger visits the cat. The tiger visits the mouse. If something likes the bald eagle then it is blue. If something visits the bald eagle and it visits the cat then the bald eagle is red. If something chases the mouse then it visits the cat. If something is blue then it chases the tiger. If something visits the cat and the cat chases the tiger then the tiger likes the bald eagle. If something likes the tiger then the tiger likes the bald eagle. If something chases the mouse then it visits the mouse.

Question:
Based on the above information, is the following statement true, false, or unknown?
The cat does not like the mouse.

Answer: False

PrOntoQA
Example: ProntoQA Q3

Problem:
Vumpuses are floral. Vumpuses are tumpuses. Tumpuses are brown. Each tumpus is a wumpus. Wumpuses are small. Each wumpus is a rompus. Each zumpus is metallic. Every rompus is happy. Rompuses are impuses. Each impus is amenable. Each impus is a dumpus. Every dumpus is not metallic. Dumpuses are numpuses. Each numpus is bitter. Each numpus is a jompus. Every jompus is cold. Each jompus is a yumpus. Wren is a tumpus. Question:
Is the following statement true or false?
Wren is not metallic.

Answer: True

FOLIO
Example: FOLIO dev Q1

Problem:
If people perform in school talent shows often, then they attend and are very engaged with school events. People either perform in school talent shows often or are inactive and disinterested members of their community. If people chaperone high school dances, then they are not students who attend the school. All people who are inactive and disinterested members of their community chaperone high school dances. All young children and teenagers who wish to further their academic careers and educational opportunities are students who attend the school. Bonnie either

both attends and is very engaged with school events and is a student who attends the school, or she neither attends and is very engaged with school events nor is a student who attends the school.

Question:
Based on the above information, is the following statement true, false, or uncertain?
If Bonnie is either both a young child or teenager who wishes to further her academic career and educational opportunities and chaperones high school dances or neither is a young child nor teenager who wishes to further her academic career and educational opportunities, then Bonnie is either a student who attends the school or is an inactive and disinterested member of the community.

Answer: True

A.2 Prompts

ProofWriter Prompts for Z3 Solver One-shot demonstration

Given a problem description and a question. The task is to parse the problem and the question into Python Z3 solver.

———

Problem:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
Question:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
###
# solution in Python:
def solution():
# Anne is quiet.
quiet(Anne)
# Erin is furry.
furry(Erin)
# Erin is green.
green(Erin)
# Fiona is furry.
furry(Fiona)
# Fiona is quiet.
quiet(Fiona)
# Fiona is red.
red(Fiona)
# Fiona is rough.
rough(Fiona)
# Fiona is white.
white(Fiona)
# Harry is furry.
furry(Harry)
# Harry is quiet.

quiet(Harry)
# Harry is white.
white(Harry)
# Young people are furry.
ForAll([x], Implies(young(x), furry(x)))
# If Anne is quiet then Anne is red.
Implies(quiet(Anne), red(Anne))
# Young, green people are rough.
ForAll([x], Implies(And(young(x), green(x)), rough(x)))
# If someone is green then they are white.
ForAll([x], Implies(green(x), white(x)))
# If someone is furry and quiet then they are white.
ForAll([x], Implies(And(furry(x), quiet(x)), white(x)))
# If someone is young and white then they are rough.
ForAll([x], Implies(And(young(x), white(x)), rough(x)))
# All red people are young.
ForAll([x], Implies(red(x), young(x)))
# Question: the following statement true, false, or unknown? Anne is white.
return white(Anne)

ProofWriter Prompts for Prover9 One shot demonstration for LLM

Given a problem description and a question, the task is to parse the problem and the question into first-order logic formulas. The grammar of the first-order logic formula is defined as follows:

1. Logical conjunction of expr1 and expr2: expr1 $\wedge$ expr2
2. Logical disjunction of expr1 and expr2: expr1 $\vee$ expr2
3. Logical exclusive disjunction of expr1 and expr2: expr1 $\oplus$ expr2
4. Logical negation of expr1: $\neg$expr1
5. expr1 implies expr2: expr1 $\rightarrow$ expr2
6. expr1 if and only if expr2: expr1 $\leftrightarrow$ expr2
7. Logical universal quantification: $\forall x$
8. Logical existential quantification: $\exists x$

Problem
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
Question:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
###
Predicates
quiet(x) ::: x is quiet.
furry(x) ::: x is furry.

green(x) ::: x is green.
red(x) ::: x is red.
rough(x) ::: x is rough.
white(x) ::: x is white.
young(x) ::: x is young
Premises
quiet(Anne) ::: Anne is quiet.
furry(Erin) ::: Erin is furry.
green(Erin) ::: Erin is green.
furry(Fiona) ::: Fiona is furry.
quiet(Fiona) ::: Fiona is quiet.
red(Fiona) ::: Fiona is red.
rough(Fiona) ::: Fiona is rough.
white(Fiona) ::: Fiona is white.
furry(Harry) ::: Harry is furry.
quiet(Harry) ::: Harry is quiet.
white(Harry) ::: Harry is white.
$\forall x(\text{young}(x) \rightarrow \text{furry}(x))$ ::: Young people are furry.
$(\text{quiet}(\text{Anne}) \rightarrow \text{red}(\text{Anne}))$ ::: If Anne is quiet then Anne is red.
$\forall x(\text{young}(x) \wedge \text{green}(x) \rightarrow \text{rough}(x))$ ::: Young, green people are rough.
$\forall x(\text{green}(x) \rightarrow \text{white}(x))$ ::: If someone is green then they are white.
$\forall x((\text{furry}(x) \wedge \text{quiet}(x)) \rightarrow \text{white}(x))$ ::: If someone is furry and quiet then they are white.
$\forall x((\text{young}(x) \wedge \text{white}(x)) \rightarrow \text{rough}(x))$ ::: If someone is young and white then they are rough.
$\forall x(\text{red}(x) \rightarrow \text{young}(x))$ ::: All red people are young.
Conclusion:
white(Anne)

ProofWriter Prompts for Pyke Solver One-shot demonstration

Task Description: You are given a problem description and a question. The task is to:
1) define all the predicates in the problem
2) parse the problem into logic rules based on the defined predicates
3) write all the facts mentioned in the problem
4) parse the question into the logic form
Problem:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
Question:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
###
Predicates:
quiet($x, bool) ::: x is quiet.
furry($x, bool) ::: x is furry.
green($x, bool) ::: x is green.
red($x, bool) ::: x is red.
rough($x, bool) ::: x is rough.
white($x, bool) ::: x is white.

young($x, bool) ::: x is young
Facts:
quiet(Anne, True) ::: Anne is quiet.
furry(Erin, True) ::: Erin is furry.
green(Erin, True) ::: Erin is green.
furry(Fiona, True) ::: Fiona is furry.
quiet(Fiona, True) ::: Fiona is quiet.
red(Fiona, True) ::: Fiona is red.
rough(Fiona, True) ::: Fiona is rough.
white(Fiona, True) ::: Fiona is white.
furry(Harry, True) ::: Harry is furry.
quiet(Harry, True) ::: Harry is quiet.
white(Harry, True) ::: Harry is white.
young($x, True) >>> furry($x, True)) ::: Young people are furry.
quiet(Anne, True) >>> red(Anne, True)) ::: If Anne is quiet then Anne is red.
young($x, True) && green($x, True) >>> rough($x, True) ::: Young, green people are rough.
green($x, True) >>> white($x, True) ::: If someone is green then they are white.
furry($x, True) && quiet($x, True) >>> white($x, True)
::: If someone is furry and quiet then they are white.
young($x, True) && white($x, True) >>> rough($x, True)
::: If someone is young and white then they are rough.
red($x, True) >>> young($x, True) ::: All red people are young.
Query:
white(Anne)

A.3 Incorrect Example Generation

The following section includes classic Incorrect translations, more incorrect translations can be found in Processed_Datasets in https://github.com/Mattylam/Logic_Symbolic_Solvers_Experiment

Example 1: Prover9 PrOntoQA_2 Parse error

.....
∀x(Dumpus(x) → Happy(x)) ::: Every dumpus is happy.
∀x(Dumpus(x) → Vumpus(x)) ::: Each dumpus is a vumpus.
∀x(Vumpus(x) → Bright(x)) ::: Vumpuses are bright.
∀x(Vumpus(x) → Jompus(x)) ::: Every vumpus is a jompus.
∀x(Jompus(x) → Large(x)) ::: Jompuses are large.
Stella is a yumpus.
Conclusion:
Bright(Stella) ::: Stella is bright.

Analysis: LLM failed to perform all translations, "Stella is yumpus" was not translated, thus system throws a parse error

Example 2: Z3 FOLIO_dev_27 Incorrect Interpretation

Problem: All aliens are extraterrestrial. If someone is from Mars, then they are aliens. No extraterrestrial is human. Everyone from Earth is a human. Marvin cannot be from Earth and from Mars. If Marvin is not from Earth, then Marvin is an extraterrestrial.

Question: Based on the above information, is the following statement true, false, or uncertain? Marvin is an alien.
Answer: False
Predicted Answer: Unknown
Translation from LLM:
.....
# All aliens are extraterrestrial.
ForAll([x], Implies(Alien(x), Extraterrestrial(x)))
# If someone is from Mars, then they are aliens.
ForAll([x], Implies(FromMars(x), Alien(x)))
# No extraterrestrial is human.
ForAll([x], Implies(Extraterrestrial(x), Not(Human(x))))
# Everyone from Earth is a human.
ForAll([x], Implies(FromEarth(x), Human(x)))
# Marvin cannot be from Earth and from Mars.
Not(And(FromEarth(marvin), FromMars(marvin)))
# If Marvin is not from Earth, then Marvin is an extraterrestrial.
Implies(Not(FromEarth(marvin)), Extraterrestrial(marvin))
# Question: Marvin is an alien.
return Alien(marvin)

Analysis: LLM incorrectly interpreted the logic of the statement. The correct translation should be Xor(And(FromEarth(marvin), FromMars(marvin))), the wording of the statement makes it ambiguous for LLMs to interpret. If the statement was, Marvin is either from Earth or from Mars, LLMs would have been able to translate it.

Example 3: Z3 FOLIO_dev_16 Difficult predicate extraction
Problem: All rabbits are cute. Some turtles exist. An animal is either a rabbit or a squirrel. If something is skittish, then it is not still. All squirrels are skittish. Rock is still.
Question: Based on the above information, is the following statement true, false, or uncertain? Rock is a turtle. Answer: Unknown
Error: ExecutionError NameError: name 'Exist' is not defined. Did you mean: 'exit'?"
Translation from LLM:
.....
All rabbits are cute.
ForAll([x], Implies(Rabbit(x), Cute(x)))
# Some turtles exist.
Exist([x], Turtle(x))
# An animal is either a rabbit or a squirrel.
ForAll([x], Or(Rabbit(x), Squirrel(x)))
# If something is skittish, then it is not still.
ForAll([x], Implies(Skittish(x), Not(Still(x))))
# All squirrels are skittish.
ForAll([x], Implies(Squirrel(x), Skittish(x)))
# Rock is still.
Still(Rock)
# Question: Rock is a turtle.
return Turtle(Rock)

A.4 How Z3 tackles BoardgameQA

A few players are playing a boardgame. The current state of the game is as follows. The cat is named Meadow. The cat published a high-quality paper. The gecko winks at the eagle. The puffin is named Peddi. And the rules of the game are as follows. Rule1: Regarding the cat, if it has a high-quality paper, then we can conclude that it prepares armor for the canary. Rule2: If you are positive that you saw one of the animals rolls the dice for the eagle, you can be certain that it will also proceed to the spot right after the sun bear. Rule3: If the cat prepares armor for the canary and the cricket gives a magnifier to the canary, then the canary will not remove one of the pieces of the aardvark. Rule4: If the cat has a name whose first letter is the same as the first letter of the puffin's name, then the cat does not prepare armor for the canary. Rule5: The canary removes one of the pieces of the aardvark whenever at least one animal proceeds to the spot right after the sun bear. Rule6: Regarding the cat, if it has something to sit on, then we can conclude that it does not prepare armor for the canary. Rule3 is preferred over Rule5. Rule4 is preferred over Rule1. Rule6 is preferred over Rule1.
Based on the game state and the rules and preferences, does the canary remove from the board one of the pieces of the aardvark?

# solution in Python:
def solution():
# rules
# Rule1: Regarding the cat, if it has a high-quality paper, then we can conclude that it prepares armor for the canary.
Rule1 = Implies(has_high_quality_paper(cat), prepare_armor(cat, canary))
# Rule2: If you are positive that you saw one of the animals rolls the dice for the eagle, you can be certain that it will also proceed to the spot right after the sun bear.
Rule2 = ForAll([x], Implies(roll_dice(x, eagle), proceed_to_spot_right_after(x, sun_bear)))
# Rule3: If the cat prepares armor for the canary and the cricket gives a magnifier to the canary, then the canary will not remove one of the pieces of the aardvark.
Rule3 = Implies(And(prepare_armor(cat, canary), give_magnifier(cricket, canary)), Not(remove_piece(canary, aardvark)))
# Rule4: If the cat has a name whose first letter is the same as the first letter of the puffin's name, then the cat does not prepare armor for the canary.
Rule4 = Implies(has_same_first_letter_name(cat, puffin), Not(prepare_armor(cat, canary)))
# Rule5: The canary removes one of the pieces of the aardvark whenever at least one animal proceeds to the spot right after the sun bear.
Rule5 = Implies(Exists([x], proceed_to_spot_right_after(x, sun_bear)), remove_piece(canary, aardvark))
# Rule6: Regarding the cat, if it has something to sit on, then we can conclude that it does not prepare armor for the canary.
Rule6 = Implies(has_something_to_sit_on(cat), Not(prepare_armor(cat, canary)))
# The current state of the game is as follows. The cat is named Meadow. The cat published a high-quality paper. The gecko winks at the eagle. The puffin is named Peddi.
# The cat is named Meadow. The puffin is named Peddi. The first letter of Meadow is M. The first letter of the Peddi is P. So the cat does not have the same first letter name as the puffin.
has_same_first_letter_name(cat, puffin) == False
The cat published a high-quality paper.
has_high_quality_paper(cat) == True
The gecko winks at the eagle.
winks_at(gecko, eagle) == True
preferences. Rule3 is preferred over Rule5. Rule4 is preferred over Rule1. Rule6 is preferred

over Rule1.
soft_rules = [Rule5, Rule1, Rule1]
# Rule3 is preferred over Rule5. So Rule5 is suppressed by the precondition of Rule3.
Rule5 = Or(And(prepare_armor(cat, canary), give_magnifier(cricket, canary)), Rule5)
# Rule4 is preferred over Rule1. So Rule1 is suppressed by the precondition of Rule4.
Rule1 = Or(has_same_first_letter_name(cat, puffin), Rule1)
# Rule6 is preferred over Rule1. So Rule1 is suppressed by the precondition of Rule6.
Rule1 = Or(has_something_to_sit_on(cat), Rule1)
# question: does the canary remove from the board one of the pieces of the aardvark?
return remove_piece(canary, aardvark)

A.5 GPT4o and Cohere command-r-plus Prompts

The prompts require some adjustments for GPT-4O and Cohere, as both models tend to produce complete executable code rather than adhering to the provided example. For instance, GPT-4O will define "s.solver()" and create the decision rule for Z3, instead of generating translations as specified in the prompt. Here we provide an overview of what is changed in the prompt.

ProofWriter GPT4O Prompts for Z3 Solver One-shot demonstration

The grammar of the first-order logic formula is defined as follows:
1) logical conjunction of expr1 and expr2: And(expr1, expr2)
2) logical disjunction of expr1 and expr2: Or(expr1, expr2)
3) logical exclusive disjunction of expr1 and expr2: Xor(expr1, expr2)
4) logical negation of expr1: Not(expr1)
5) expr1 implies expr2: Implies(expr1, expr2)
6) expr1 if and only if expr2: expr1 == expr2
7) logical universal quantification: ForAll()
8) logical existential quantification: Exists()
Given a problem description and a question. The task is to parse the [Problem] and the [Question] into Python Z3 solver. You are meant to follow the example format and do not provide any further explanations. Keep all the # signs as symbols and do not interpret them as markdown marker.
—
[Problem]:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
[Question]:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
####
[Problem Parse Output]:
# Anne is quiet.
quiet(Anne)
# Erin is furry.
furry(Erin)
# Erin is green.
green(Erin)
# Fiona is furry.

```
furry(Fiona)
# Fiona is quiet.
quiet(Fiona)
# Fiona is red.
red(Fiona)
# Fiona is rough.
rough(Fiona)
# Fiona is white.
white(Fiona)
# Harry is furry.
furry(Harry)
# Harry is quiet.
quiet(Harry)
# Harry is white.
white(Harry)
# Young people are furry.
ForAll([x], Implies(young(x), furry(x)))
# If Anne is quiet then Anne is red.
Implies(quiet(Anne), red(Anne))
# Young, green people are rough.
ForAll([x], Implies(And(young(x), green(x)), rough(x)))
# If someone is green then they are white.
ForAll([x], Implies(green(x), white(x)))
# If someone is furry and quiet then they are white.
ForAll([x], Implies(And(furry(x), quiet(x)), white(x)))
# If someone is young and white then they are rough.
ForAll([x], Implies(And(young(x), white(x)), rough(x)))
# All red people are young.
ForAll([x], Implies(red(x), young(x)))
[Question Parse Output]:
# Question: the following statement true, false, or unknown? Anne is white.
return white(Anne)
```

ProofWriter Cohere Prompts for Z3 Solver One-shot demonstration

For the Z3 solver, the Cohere prompt was slightly adjusted because produces translation not aligned with the given example.

```
The grammar of the first-order logic formula is defined as follows:
1) logical conjunction of expr1 and expr2: And(expr1, expr2)
2) logical disjunction of expr1 and expr2: Or(expr1, expr2)
3) logical exclusive disjunction of expr1 and expr2: Xor(expr1, expr2)
4) logical negation of expr1: Not(expr1)
5) expr1 implies expr2: Implies(expr1, expr2)
6) expr1 if and only if expr2: expr1 == expr2
7) logical universal quantification: ForAll()
8) logical existential quantification: Exists()
Given a problem description and a question. The task is to parse the [Problem] and the [Question] into Python Z3 solver. You are meant to follow the example format and do not provide any further explanations. Follow the format given and do not define "s" and "s.solver" for the Z3 solver. Keep all the # signs as symbols and do not interpret them as markdown marker.
```

——
[Problem]:
Anne is quiet.
.....

ProofWriter GPT4o and Cohere Prompts for Prover9 One shot demonstration for LLM

The grammar of the first-order logic formula is defined as follows:

1. Logical conjunction of expr1 and expr2: expr1 $\wedge$ expr2
2. Logical disjunction of expr1 and expr2: expr1 $\vee$ expr2
3. Logical exclusive disjunction of expr1 and expr2: expr1 $\oplus$ expr2
4. Logical negation of expr1: $\neg$expr1
5. expr1 implies expr2: expr1 $\rightarrow$ expr2
6. expr1 if and only if expr2: expr1 $\leftrightarrow$ expr2
7. Logical universal quantification: $\forall x$
8. Logical existential quantification: $\exists x$

Given a problem description and a question. The task is to parse the [Problem] and the [Question] into Prover9 solver. You are meant to follow the example format and do not provide any further explanations. Keep all the ::: signs as symbols and do not interpret them as markdown marker.
[Problem]:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
[Question]:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
####
[Problem Parse Output]:
Predicates
quiet(x) ::: x is quiet.
furry(x) ::: x is furry.
green(x) ::: x is green.
red(x) ::: x is red.
rough(x) ::: x is rough.
white(x) ::: x is white.
young(x) ::: x is young
Premises
quiet(Anne) ::: Anne is quiet.
furry(Erin) ::: Erin is furry.
green(Erin) ::: Erin is green.
furry(Fiona) ::: Fiona is furry.
quiet(Fiona) ::: Fiona is quiet.
red(Fiona) ::: Fiona is red.
rough(Fiona) ::: Fiona is rough.

white(Fiona) ::: Fiona is white.
furry(Harry) ::: Harry is furry.
quiet(Harry) ::: Harry is quiet.
white(Harry) ::: Harry is white.
$\forall x(\text{young}(x) \rightarrow \text{furry}(x))$ ::: Young people are furry.
$(\text{quiet(Anne)} \rightarrow \text{red(Anne)})$ ::: If Anne is quiet then Anne is red.
$\forall x(\text{young}(x) \wedge \text{green}(x) \rightarrow \text{rough}(x))$ ::: Young, green people are rough.
$\forall x(\text{green}(x) \rightarrow \text{white}(x))$ ::: If someone is green then they are white.
$\forall x((\text{furry}(x) \wedge \text{quiet}(x)) \rightarrow \text{white}(x))$ ::: If someone is furry and quiet then they are white.
$\forall x((\text{young}(x) \wedge \text{white}(x)) \rightarrow \text{rough}(x))$ ::: If someone is young and white then they are rough.
$\forall x(\text{red}(x) \rightarrow \text{young}(x))$ ::: All red people are young.
[Question Parse Output]:
Conclusion:
white(Anne)

ProofWriter GPT4o and Cohere Prompts for Pyke Solver One-shot demonstration

The grammar of the first-order logic formula is defined as follows:
1) logical conjunction of expr1 and expr2: expr1 && expr2
2) logical negation of expr1: expr1($x, False), as example if "Anne is not quiet", the term would be "Quiet(Anne, False)"
3) expr1 implies expr2: expr1 »> expr2
Given a problem description and a question. The task is to parse the [Problem] and the [Question] into Pyke solver. You are meant to follow the example format and do not provide any further explanations. Keep all the ::: signs as symbols and do not interpret them as markdown marker.
[Problem]:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.
[Question]:
Based on the above information, is the following statement true, false, or unknown? Anne is white.
####
[Problem Parse Output]:
Predicates:
quiet($x, bool) ::: x is quiet.
furry($x, bool) ::: x is furry.
green($x, bool) ::: x is green.
red($x, bool) ::: x is red.
rough($x, bool) ::: x is rough.
white($x, bool) ::: x is white.
young($x, bool) ::: x is young
Facts:
quiet(Anne, True) ::: Anne is quiet.
furry(Erin, True) ::: Erin is furry.
green(Erin, True) ::: Erin is green.
furry(Fiona, True) ::: Fiona is furry.
quiet(Fiona, True) ::: Fiona is quiet.
red(Fiona, True) ::: Fiona is red.

rough(Fiona, True) ::: Fiona is rough.
white(Fiona, True) ::: Fiona is white.
furry(Harry, True) ::: Harry is furry.
quiet(Harry, True) ::: Harry is quiet.
white(Harry, True) ::: Harry is white.
young($x, True) >>> furry($x, True)) ::: Young people are furry.
quiet(Anne, True) >>> red(Anne, True)) ::: If Anne is quiet then Anne is red.
young($x, True) && green($x, True) >>> rough($x, True) ::: Young, green people are rough.
green($x, True) >>> white($x, True) ::: If someone is green then they are white.
furry($x, True) && quiet($x, True) >>> white($x, True)
::: If someone is furry and quiet then they are white.
young($x, True) && white($x, True) >>> rough($x, True)
::: If someone is young and white then they are rough.
red($x, True) >>> young($x, True) ::: All red people are young.
[Question Parse Output]:
Query:
white(Anne)

Dataset	**Z3**	**Prover9**	**Pyke**
	Avg_Acc	Avg_Acc	Avg_Acc
ProofWriter D5 OWA	85.75%	75.00%	56.63%
ProofWriter D3 OWA	83.04%	75.37%	52.37%
ProofWriter D2 OWA	82.50%	76.00%	56.50%
ProofWriter D5 CWA	87.50%	78.25%	60.25%
ProofWriter D3 CWA	89.25%	76.13%	45.63%
ProofWriter D2 CWA	89.63%	77.12%	54.75%
PrOntoQA	94.12%	76.87%	91.12%
FOLIO (1 Shot)	26.25%	43.78%	✗
FOLIO (2 Shot)	34.36%	41.60%	✗
FOLIO (4 Shot)	36.87%	49.13%	✗

Table 4: Average accuracy of Experiment done with GPT-4o, GPT-3.5-turbo, Gemini-1.0-pro and command-r-plus on all datasets. We present the percentage of the overall average accuracy of tools (Avg_Acc). The shots represent the number of shots used in the prompt.✗: the tool was unable to solve this dataset. The numbers highlighted in red color represent the highest accuracy between the 3 chosen tools.