# FoVer: First-Order Logic Verification for Natural Language Reasoning

**Yu Pei**[1,2] **Yongping Du**[1,2*] **Xingnan Jin**[1,2]
[1]College of Computer Science, Beijing University of Technology, Beijing, China
[2]Beijing Artificial Intelligence Institute, Beijing University of Technology, Beijing, China
ypdu@bjut.edu.cn

## Abstract

Large Language Models (LLMs) have shown remarkable capabilities in various tasks, including logical reasoning. However, their propensity for generating incorrect or inconsistent responses remains a significant concern. To address this issue, we propose FoVer (**F**irst-**o**rder logic **Ver**ification), an automated pipeline that verifies the logical correctness of reasoning texts using first-order logic. The pipeline operates in two main steps: (1) LLM-driven translation of natural language into executable logical expressions, and (2) automated logical verification using the Z3 theorem prover. We evaluate FoVer on specialized logical datasets (ProofWriter and FOLIO) and real-world LLM outputs (REVEAL). The results demonstrate that FoVer outperforms existing methods in logical verification significantly, showing notable improvements in accuracy across both ideal and practical scenarios. The pipeline also demonstrates its potential in identifying annotation errors in existing datasets, and it could be utilized for constructing new logical reasoning datasets. This work presents a significant step forward in enhancing the trustworthiness of LLM outputs, particularly in tasks requiring logical integrity.[1]

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable progress in recent years, with their impressive capabilities on various tasks (Brown et al., 2020; OpenAI et al., 2024; Naveed et al., 2024; Zhao et al., 2024). However, despite their significant advancements, LLMs still face challenges in terms of reliability and accuracy, particularly in the realm of factual consistency and logical reasoning (Wei et al., 2022; Lyu et al., 2023; Pan et al., 2023; Chen et al., 2024).

One of the most pressing concerns surrounding LLMs is their propensity to generate incorrect responses, particularly in the form of ‘‘hallucinations’’—generating content that appears plausible but is factually incorrect or logically inconsistent (Lin et al., 2022; Maynez et al., 2020). This issue undermines the trustworthiness of these models and limits their applicability in critical domains where accuracy is paramount.

Researchers have developed various methods to enhance the logical reasoning capabilities of LLMs (Kojima et al., 2022; Zhou et al., 2022; Chen et al., 2024). One of them is Chain-of-Thought (CoT) prompting (Wei et al., 2022), which uses well-designed prompts to elicit the models to generate step-by-step explanations, thereby improving the accuracy of their logical inferences. Notably, some LLM providers combine such thinking process with test-time compute (Snell et al., 2025), releasing more powerful models, including OpenAI o1 (OpenAI, 2024), DeepSeek R1 (DeepSeek-AI et al., 2025), and Qwen-QwQ (Qwen Team, 2024). While these techniques enhance the performance of LLMs on complex reasoning tasks, issues of factuality and logical correctness persist. Specifically, when generating step-by-step reasoning, they may produce incorrect or incomplete or redundant rationales, or make mistakes during reasoning processes (Chen et al., 2024). Figure 1a presents an example of an incorrect CoT answer where premises cannot draw the conclusion.

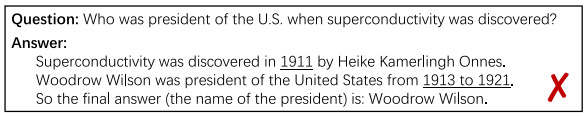
To improve trustworthiness of LLM outputs, several studies adopt automatic methods to verify the quality of the reasoning claims (Jacovi et al., 2024; Opitz and Frank, 2021; Golovneva et al., 2022; Zong and Lin, 2024). Some attempts incorporate external theorem provers as a means of reasoning or logical verification (Olausson et al., 2023; Pan et al., 2023; Lalwani et al., 2024). External theorem provers such as Prover9 and Z3 (de Moura and Bjørner, 2008) employ formal logical calculus to verify mathematical statements

*Corresponding author.

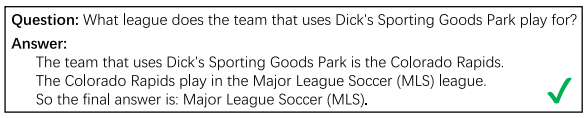
[1]Prompt and code are available at https://github.com/peiyu-cn/FoVer.



Transactions of the Association for Computational Linguistics, vol. 13, pp. 1340–1359, 2025. https://doi.org/10.1162/TACL.a.41
Action Editor: Katrin Elisabeth Erk. Submission batch: 10/2024; Revision batch: 2/2025; Published 10/2025.




(a) Logically incorrect CoT answer (from Press et al. (2023))




(b) Correct CoT answer (from REVEAL-MuSiQue)

Figure 1: Examples of CoT answer. The proposed method can examine logical coherence in such cases.

and logical arguments rigorously and automatically. Their deterministic nature ensures reliability in logical reasoning tasks. However, current approaches still primarily focus on coarse-grained and inflexible logic forms and are unable to find implications from text, hindering their applicability to real-world LLM outputs. Moreover, they often underutilize the vast knowledge acquired by LLMs during pre-training.

To address these limitations, we propose FoVer (**F**irst-**o**rder logic **Ver**ification), an automated pipeline to verify the **logical correctness** of reasoning texts. FoVer operates by *first* using an LLM to translate natural language reasoning via CoT prompting into an executable first-order logical (FOL) format, which is *then* verified by an external theorem prover rigorously. This design addresses the two challenges: the **limited applicability** of existing methods integrating theorem provers, and the inherent **trustworthiness concerns** of LLM outputs. By leveraging LLMs' text comprehension capabilities and world knowledge, FoVer can extract fine-grained logical information from diverse real-world texts, extending verification beyond specialized logical datasets. Meanwhile, the integration of external theorem provers provides rigorous logical validation, ensuring the reliability of the output.

By bridging the gap between natural language understanding and rigorous logical analysis, FoVer provides a generalizable credible mechanism for validating the logical consistency of LLM outputs. This validation process helps identify potential logical flaws in complex reasoning chains produced by LLMs, thereby enhancing the trustworthiness of LLM applications in critical reasoning tasks.

## 2 Related Work

### 2.1 Logical Reasoning in Language Models

Recent works explore different approaches to enhance the logical reasoning capabilities of LLMs. One line of research focuses on fine-tuning LLMs on carefully curated datasets containing logical problems (Liu et al., 2020; Talmor et al., 2019; Liu et al., 2023). However, these fine-tuning approaches typically require substantial computational resources and large-scale training data (Zhao et al., 2024; Brown et al., 2020; Han et al., 2024b). Moreover, as the enhancement occurs during the training stage, the reasoning process remains opaque, making it difficult to understand or verify the models' decision-making process, potentially leading to reliability concerns.

Wei et al. (2022) introduce CoT prompting to guide LLM reasoning step by step, enabling models to break down complex problems into intermediate steps. Following this approach, various prompting methods have been developed (Kojima et al., 2022; Press et al., 2023; Zhou et al., 2022). These approaches, while requiring no additional training, have demonstrated remarkable effectiveness in enhancing LLMs' reasoning capabilities. Importantly, they provide a certain degree of interpretability by making the reasoning process explicit and observable, allowing for better understanding and verification of the models' logical deduction steps.

### 2.2 Theorem Prover

Automated Theorem Proving (ATP) has emerged as a critical technology in formal verification, enabling rigorous reasoning about mathematical statements and program properties. Notably, Satisfiability Modulo Theories (SMT) solvers, while primarily designed for deciding satisfiability of formulas in specific theories, have demonstrated capabilities in handling traditional theorem proving tasks. Modern SMT solvers such as Z3 (de Moura and Bjørner, 2008) and CVC5 (Barbosa et al., 2022) effectively combine the efficiency of SAT solving with decision procedures for various theories, making them powerful tools for both satisfiability checking and theorem proving.

There are several studies leveraging theorem provers to enhance the reasoning capabilities of language models. Olausson et al. (2023), Pan et al. (2023), and Ye et al. (2023) develop similar pipelines to ours, where language models first

translate natural language text into a specific format, which is then processed by symbolic solvers. However, Olausson et al. (2023) and Pan et al. (2023) translate natural language into a normal form that requires non-ASCII characters such as ‘‘∀’’ and ‘‘¬’’, and they do not integrate CoT prompting into their approaches, resulting in lower rate of executability. Olausson et al. (2023) employ majority voting to alleviate this issue, while Pan et al. (2023) just fall back to CoT. In a distinct work, Ye et al. (2023) use Z3 Python API as the translation target, but they focus more on solving math problems. Most importantly, while these approaches demonstrate the potential of combining LLMs with theorem provers, they can hardly verify real-world LLM outputs, which is the main objective of this work.

Meanwhile, Lalwani et al. (2024) use fine-tuned models (LogicLLaMA [Yang et al., 2024]) instead of raw LLMs to translate natural language into FOL. They also employ NLI models to involve world knowledge. However, their FOL representations have rigid syntactic structures, and NLI finetuning models do not generalize to out-of-domain complex logical structures (Jacovi et al., 2024). In contrast, our pipeline elicits LLMs to complement necessary common knowledge directly, leveraging their capabilities and knowledge acquired during pre-training.

## 3 FoVer Pipeline

### 3.1 Overview

The FoVer pipeline consists of two steps, as shown in Figure 2:

1) **LLM-driven Logic Translation**: It leverages an LLM to generate a Python function consisting of logic representations from given natural language input.
2) **Automated Logical Verification**: The translated Python function is passed to a theorem prover to execute and verify the logical validity rigorously.

Our method utilizes the Z3 Prover (de Moura and Bjørner, 2008), an SMT solver. SMT solvers are designed to determine whether a set of logical statements is satisfiable under various mathematical theories. Our goal is to verify whether the premises of an argument logically entail its conclusions. This is achieved by checking the unsatisfiability of the premises conjoined with the

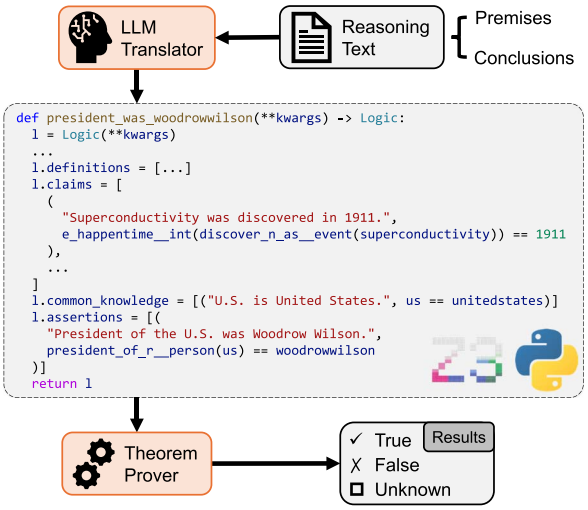



Figure 2: Overview of FoVer pipeline. Starting from a reasoning text, (1) the LLM translator identifies its premises and conclusions and translates it into a Python function consisting of first-order logic. (2) The theorem prover then processes the translated logic theory, returning whether the conclusions are entailed (True), contradicted (False), or undetermined.

negation of the conclusion. Let $p := \bigwedge_{i=1}^{n} p_i$ be premises, $q_j$ be conclusion, we want to prove

$$p \rightarrow q_j$$

which is equivalent to

$$\neg \left( p \wedge \neg q_j \right)$$

therefore,

$$p \rightarrow q_j \Leftrightarrow \text{unsat} \left( p \wedge \neg q_j \right)$$

We choose to work with the Z3 Python API, because Python code is more accessible and easier for LLMs to generate correctly, given their extensive training on programming languages.

To simplify the interaction with Z3 and reduce the cognitive load on LLMs, we develop a custom wrapper class. This class encapsulates the complexities of working directly with the Z3 solver, allowing for a more straightforward specification of premises and conclusions. In the pipeline, LLMs are instructed to generate a Python function that returns an instance of this wrapper class, which then enforces verification process.

For premises, we devise two additional parts besides those explicit statements made in text. The first part is definitions, where relations and constraints in the reasoning process are formalized, specifying how different predicates should

be interpreted and what properties they possess (e.g., constraints for an injective relation). The second part is common knowledge, where LLMs could add any unstated assumption they consider necessary to draw the conclusion. The devisal of common knowledge leverages world knowledge that LLMs gained during pre-training, and enables them to capture implications from the text, which other methods are incapable of doing.

The following sections describe the specifics of each step in the FoVer pipeline. Section 3.2 details our novel translation process, explaining how we prompt LLMs to generate appropriate FOL representations in Python. Section 3.3 then explores the execution and verification phase.

### 3.2 Step A: Translating Natural Language to FOL Python Function

The translation is performed in a coarse-to-fine fashion, where LLMs are instructed to think first and then produce the grounded code.

A translation thinking process is designed to guide LLMs to produce results more consistently. Each demo (In-Context example) in the few-shot prompt includes a detailed function comment, outlining the step-by-step thinking in natural language, which can be considered as Chain of Thought, shown in Figure 3. The process begins with identifying the predicates. The focus on predicates is grounded in their central role in natural language semantics (Davidson, 1966), which provides the essential building blocks for determining sorts[2] and translating statements.

In consequent function bodies, we provide thorough annotations for Python statements and a description for each Z3 expression, facilitating LLMs in translation and writing Python code more accurately. There are also several systematic coding conventions developed, illustrated in Figure 4:

- **Delayed Quantifier Declaration**. Since quantifier variables must be declared before use in Python, LLMs often struggle to predict which variables will be needed across multiple logical expressions, leading to undefined variable errors. This convention defers the variable declarations to the end of the function, which allows LLMs to first construct all logical expressions using undeclared quantifiers, then declare all used variables at the bottom (e.g., `n1, n2 = Consts('n1 n2', Newthing)`), ensuring completeness and avoiding premature commitment to specific quantifier sets.
- **Placeholder Naming Guidance**. We find that LLMs often misapply functions; they pass arguments of incorrect type or number, or use mismatched return values. To alleviate this issue and to avoid naming conflicts, we instruct models to adopt consistent and predictable identifiers.
- **Type-Encoded Identifier and Usage Guidance**. To further resolve the misapplication issue, we also develop a systematic naming convention for function identifiers, which

[2]A sort is a concept from Many-Sorted Logic (MSL), analogous to a type in programming languages; it defines a domain of values and the operations that can be performed on those values. In reasoning context, for example, ''Person'' can be the sort of entity ''William''.

**Claims:**
Superconductivity was discovered in 1911 by Heike Kamerlingh Onnes.
Woodrow Wilson was president of the United States from 1913 to 1921.

**Target:** President of the U.S. when superconductivity was discovered was Woodrow Wilson.
**Predicates:** discover in, discover by, be president of from to, be president of when.
**Parameters of predicates:**
discover in: New thing discover in Int, *-to-1, (New thing) -> Int.
- New thing
discover by: New thing discover by Person, *-to-1, (New thing) -> Person.
- Person
be president of from to: Person be president of Region from Int to Int, *-to-*, (Person, Region, Int) -> Bool. # [Preson] be president of [Region] from [Int a] to [Int b] == [Person] be president of [Region] in [Int x] (a <= x <= b), I will use this pattern for all range relations.
- Region
president of when: president of Region when Event happen is Person, *-to-1, (Region, Event) -> Person.
- Event
**All sorts by now:** New thing, Person, Region, Event.
**Concepts:** Superconductivity, 1911, Heike Kamerlingh Onnes, Woodrow Wilson, United States, 1913, 1921, U.S.
- Superconductivity: New thing
- 1911: Int
- Heike Kamerlingh Onnes: Person
- Woodrow Wilson: Person
- United States: Region
- 1913: Int
- 1921: Int
- U.S.: Region
**Rest sorts:** Event. # Sorts to which no concept above belongs.
- Event:
Concepts: # What constitutes an Event.
- discover superconductivity
Implicit predicates:
- discover: discover New thing is Event, 1-to-1, (New thing) -> Event.
**Supplemental predicates:**
- happen in: Event happen in Int, *-to-1, (Event) -> Int.
**Removed & merged predicates:**
- discover in: [New thing] discover in [Int] ==
discover [New thing] is [Event e], [Event e] happen in [Int].
**All sorts:** New thing, Person, Region, Event; Int, Bool.

Figure 3: CoT annotation within the first demo (In-Context example). The Question-Answer pair of this demo is shown in Figure 1a, function body is shown in Table 6. In FoVer step A, the LLM first outputs this CoT annotation, then generates the rest Python function body.

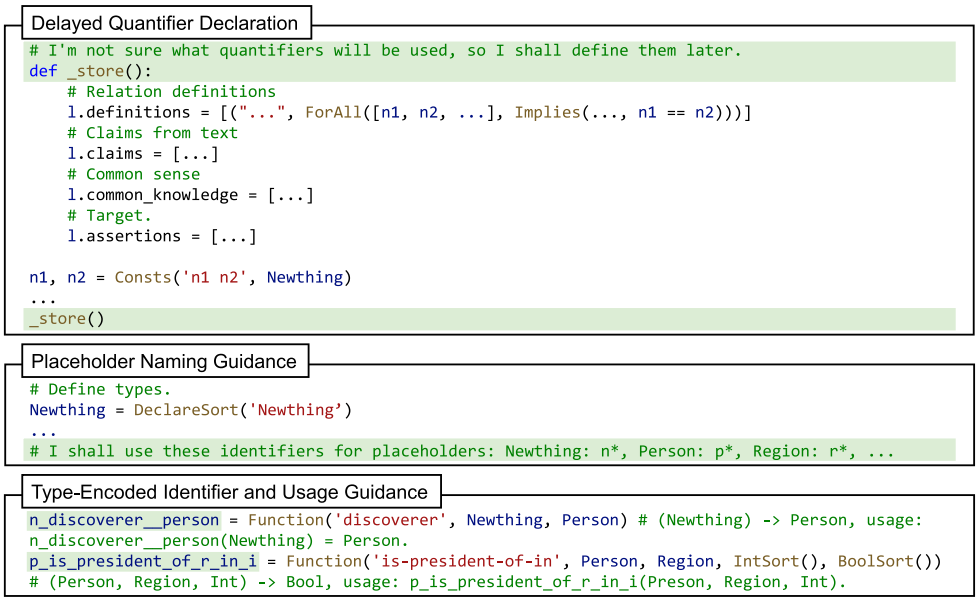



Figure 4: Coding conventions used in few-shot demos. Only the main structure is shown here. Full demo is presented in Table 6.

encodes parameter types and return type directly into the identifier. In this convention, (1) parameter types are indicated by prefixes that consistently match placeholder naming (e.g., `n...` for `Newthing`); (2) double underscores followed by an unabbreviated type name indicate a non-Boolean return type; and (3) each function definition is accompanied by an explicit usage comment, which further reduces type-related errors.

A complete example of the function body is presented in Table 6.

### 3.3 Step B: Executing Translated Function

Once the above function is generated, the Python interpreter can execute it. The verification procedure is described in Algorithm 1, which is performed by the pre-defined wrapper class automatically. To ensure verification results are dependable, additional validations are introduced. The procedure checks satisfiability of premises (sat($p$)) to detect paradoxes (step 3), where unsatisfiable premises from claims and common knowledge trigger an exception. On the other hand, when definitions encounter unsatisfiability, the wrapper class removes the unsatisfiable core to obtain the maximal satisfiable subset of definitions. This treatment respects the auxiliary nature of definitions, which serve to enforce additional constraints (such as one-to-one relation

**Algorithm 1:** Verification Procedure

**Input:** Set of claims $\mathcal{C}$, set of definitions $\mathcal{D}$, set of common knowledge $\mathcal{K}$ (when using), target conclusions $\boldsymbol{q}_{n\times 1}$.
**Output:** Verification results $\boldsymbol{R}_{n\times 2}$.

**Initialize** empty Z3 constraint set $\mathcal{P}$.
$\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{C}$.
**if** unsat($\mathcal{P}$) **then**
  **raise** error (paradox claims).
**end**
$\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{D}$.
**if** unsat($\mathcal{P}$) **then**
  $\mathcal{D}_0 \leftarrow$ unsat_core($\mathcal{P}$) $\cap\, \mathcal{D}$.
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{D}_0$.
**end**
$\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{K}$.
**if** unsat($\mathcal{P}$) **then**
  **raise** error (conflict common knowledge).
**end**
**foreach** $q_j$ in $\boldsymbol{q}$ **do**
  **if** $q_j \in \mathcal{D} \cup \mathcal{K}$ **then**
    **raise** error (cheating).[3]
  **end**
  **if** $q_j \in \{\text{True}, \text{False}\}$ **then**
    **raise** error.
  **end**
  $r_1 \leftarrow$ check $(\mathcal{P} \cup \{q_j\})$.
  $r_2 \leftarrow$ check $(\mathcal{P} \cup \{\neg q_j\})$.
  **yield** $(r_1, r_2)$.
**end**

constraint)—if such auxiliary leads to contradictions, they are problematic and need to be excluded. Another important validation is to check whether the target conclusion is included directly

[3]No such instance is found in experiments.

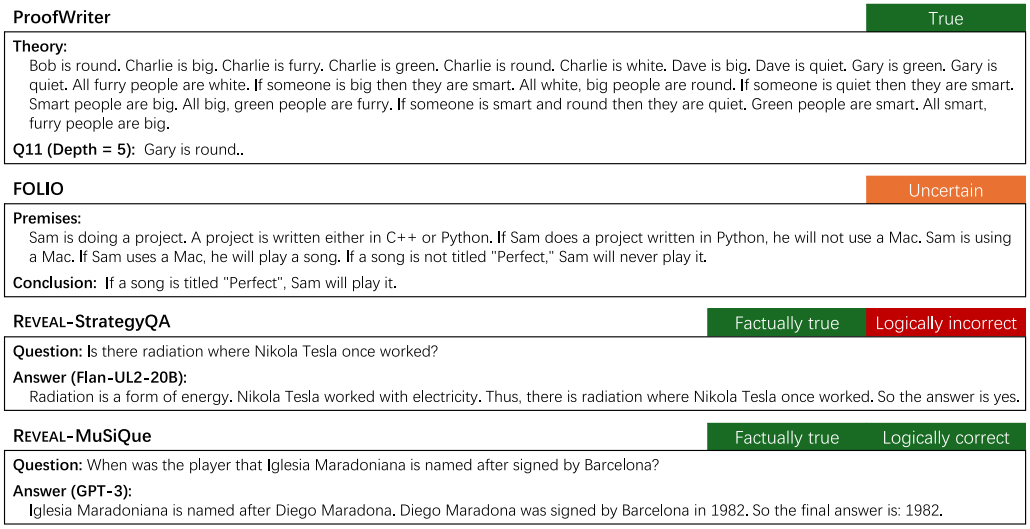



Figure 5: Examples from each dataset with their corresponding labels. Note that the REVEAL dataset contains both factual and logical labels, but only logical labels are used in experiments.

in definitions or common knowledge, and whether it is True or False literally, to avoid LLMs from cheating (step 16 and 19). The final phase checks the satisfiability of $p \wedge q_j$ (step 22) and $p \wedge \neg q_j$ (step 23), respectively, and yields the verification result.

## 4 Experiment Setup

The evaluation of FoVer follows a two-stage design to ensure both its logical verification capability and real-world applicability. In the first stage, we validate the foundational capability of FoVer using specialized logical datasets, this serves as a prerequisite verification of the method itself. Building upon these validation results, the second stage examines FoVer's effectiveness in verifying real-world LLM outputs, which directly addresses our primary research objective.

### 4.1 Datasets

Three datasets are used in experiments, as illustrated in Figure 5. To reduce computational and economic costs, we sample approximately 120–200 examples from each dataset. Despite the relatively small sample size, the results can remain insightful, as discussed in Press et al. (2024). Detailed descriptions of these datasets are provided below.

**ProofWriter** (Tafjord et al., 2021) is a dataset containing numerous rulebases of facts and rules expressed in English. Each rulebase includes a set of questions that can be proven True, False, or Unknown using proofs of various depths. There are two versions, CWA (Closed-world-assumption) and OWA (Open-world-assumption), both containing five subsets, each with different maximum reasoning depth (0, 1, 2, 3, 5). We evaluate our method on the hardest OWA-5 subset, with 200 samples selected from the test split. For each sample, one question is selected randomly (a total of 10–20 questions for each sample) with a ratio of 2:1 between depth-5 and those of Unknown, leading to an approximately balanced distribution of True/ False/Unknown labels.

**FOLIO** (Han et al., 2024a) is a human-annotated, logically complex, and diverse dataset designed for natural language reasoning with FOL. Compared to ProofWriter, premises and conclusions in FOLIO are expressed more naturally. We use all 203 samples of the validation split (the test split is not publicly available).

**REVEAL** (Jacovi et al., 2024) is a benchmark dataset for verifying LLM reasoning chains. It includes CoT output of three LLMs on four popular QA datasets: Fermi (Kalyan et al., 2021), MuSiQue (Trivedi et al., 2022), Sports Understanding (Srivastava et al., 2023), and StrategyQA (Geva et al., 2021). This dataset features real imbalanced data annotated by human developers, focusing on both factuality and logical correctness. It is prominent due to its use of **real-world data** rather than synthetic or semi-synthetic data. We randomly select 120 questions from MuSiQue and StrategyQA respectively, using one model-generated answer per question. For our focus on logic validation, only the logical labels are used.

| Dataset | GPT-4o | | | | Claude 3.5 Sonnet | | | |
|---|---|---|---|---|---|---|---|---|
| | CoT | Logic-LM | FoVer (−c) | FoVer (+c) | CoT | Logic-LM | FoVer (−c) | FoVer (+c) |
| ProofWriter | 43.5% | 37.0% | **95.5%**$^{*\dagger}$ | 89.5%$^{*\dagger}$ | 54.5% | 43.5%$^{*}$ | **94.0%**$^{*\dagger}$ | 91.0%$^{*\dagger}$ |
| FOLIO | **72.4%** | 62.1%$^{*}$ | 70.4%$^{\dagger}$ | 70.9%$^{\dagger}$ | 75.4% | 54.7%$^{*}$ | 71.9%$^{\dagger}$ | **75.9%**$^{\dagger}$ |
| REVEAL-StrategyQA | **79.2%** | / | 50.8%$^{*}$ | 59.2%$^{*}$ | **75.8%** | / | 53.3%$^{*}$ | 68.3% |
| REVEAL-MuSiQue | 55.0% | / | 70.0%$^{*}$ | **74.2%**$^{*}$ | 64.2% | / | 71.7% | **78.3%**$^{*}$ |

Table 1: Accuracy of CoT[4], Logic-LM, and FoVer pipeline. Here, ''+c'' and ''−c'' denote FoVer with and without common knowledge added, respectively. ''*'' refers to statistically significant differences ($p < 0.05$) with respect to CoT in McNemar test (McNemar, 1947); ''†'' refers to statistically significant differences compared to Logic-LM.

### 4.2 Baselines

On ProofWriter and FOLIO, we compare the performance of our pipeline against two baselines, 2-shot CoT[4] and Logic-LM (Pan et al., 2023), latter of which is a previous approach incorporating theorem prover with LLMs. On REVEAL, our pipeline is compared against single decision.[4]

The proposed method as well as baselines are evaluated with two popular and powerful LLMs: GPT-4o[5] and Claude 3.5 Sonnet.[6] For reproducibility, the temperature is set to 0 and top-$p$ is set to 1 on all experiments.

## 5 Results and Analyses

Table 1 presents an overview of main results across the three datasets. For both Logic-LM and FoVer, the response may have syntax and grammar errors, in which case it would be unable to be executed. Unlike Pan et al. (2023), who adopt fallback strategies, we count these inexecutable samples as incorrect answers.

FoVer demonstrates significant performance improvements over the baselines in most scenarios, except REVEAL-StrategyQA. On ProofWriter, the accuracy reaches 95.5% and 94.0% on GPT-4o and Claude 3.5 Sonnet, respectively. It is also observed that Claude 3.5 Sonnet performs better than GPT-4o generally. In the following subsections, we provide detailed analyses of the outcomes on each dataset, elucidating the key findings of our approach.

### 5.1 ProofWriter Results

Detailed results on ProofWriter are shown in Table 2. The reason executability rates of Logic-LM are unexpectedly low is mainly because Logic-LM utilizes PyKE (Frederiksen, 2008) for ProofWriter, which does not directly support negation operator. But in many cases, negation is necessary and LLMs attempt to use negation marks such as '$\sim$' and '$\neg$', resulting in execution failure. Additionally, GPT-4o often attempts to use '$\rightarrow$' for implications, instead of '$\ggg$' presented in few-shot demos, resulting in even lower executability rate. However, whether considering only executable samples or not, FoVer significantly outperforms Logic-LM on ProofWriter.

| Method | | GPT-4o | | | Claude 3.5 Sonnet | | |
|---|---|---|---|---|---|---|---|
| | | Exe | Acc | Total | Exe | Acc | Total |
| CoT | | / | 43.5 | 43.5 | / | 54.5 | 54.5 |
| Logic-LM | | 50.5 | 73.3$^{*}$ | 37.0 | 56.0 | 77.7$^{*}$ | 43.5$^{*}$ |
| | r | 50.5 | 76.2$^{*}$ | 38.5 | 56.0 | 77.7$^{*}$ | 43.5$^{*}$ |
| FoVer | −c | **97.0**$^{\dagger}$ | 98.5$^{*\dagger}$ | 95.5$^{*\dagger}$ | **95.5**$^{\dagger}$ | 98.4$^{*\dagger}$ | 94.0$^{*\dagger}$ |
| | +c | 91.0$^{\dagger}$ | 98.4$^{*\dagger}$ | 89.5$^{*\dagger}$ | 92.5$^{\dagger}$ | 98.4$^{*\dagger}$ | 91.0$^{*\dagger}$ |
| | r, −c | **97.0**$^{\dagger}$ | **100.0**$^{*\dagger}$ | **97.0**$^{*\dagger}$ | **95.5**$^{\dagger}$ | **100.0**$^{*\dagger}$ | **95.5**$^{*\dagger}$ |
| | r, +c | 91.0$^{\dagger}$ | **100.0**$^{*\dagger}$ | 91.0$^{*\dagger}$ | 92.5$^{\dagger}$ | **100.0**$^{*\dagger}$ | 92.5$^{*\dagger}$ |
| FoVer (strict) | −c | **98.5** | 79.7 | 78.5 | **96.0** | 80.7 | 77.5 |
| | +c | 90.5 | 76.8 | 69.5 | 94.0 | 81.4 | 76.5 |
| | r, −c | **98.5** | **100.0** | **98.5** | **96.0** | **100.0** | **96.0** |
| | r, +c | 90.5 | **100.0** | 90.5 | 94.0 | **100.0** | 94.0 |

Table 2: Detailed results on ProofWriter (%). Exe is executability rate; Acc is accuracy of executable part; Total is overall accuracy, equivalent to Exe × Acc. ''r'' denotes a relaxed evaluation setting where method predictions of True and False are considered correct when ground truth is labelled Unknown (detailed in Section 5.1). ''+c'' and ''−c'' denote results with and without common knowledge added respectively. ''*'' indicates statistical significance against CoT; ''†'' indicates statistical significance against Logic-LM.

[4]The prompts are shown in Figure 12.
[5]gpt-4o-2024-08-06.
[6]claude-3-5-sonnet-20240620.

In early experiments on the development set, we serendipitously discovered that FoVer produces True/False answers to a few questions in some cases, whereas the questions are labeled as Unknown (detailed in Appendix B). Upon human analysis, FoVer's answers were proved to be correct, and the questions are indeed decidable. We also observed several cases that FoVer identifies as paradoxes, which indeed are. These two findings indicate that our method is able to identify potential problems of logical datasets, especially those generated automatically by specific rules. This side effect benefits from the nature of a deterministic theorem prover. When considering True/False results as correct in cases where ground truth labels are Unknown (denoted as ''r'' in Table 2), accuracies of both FoVer and Logic-LM increase. Notably, accuracy of the executable part of FoVer reaches 100% on both LLMs.

To better leverage the potential of FoVer, we introduce a stricter evaluation approach for it on ProofWriter (marked as ''strict'' in Table 2), where all questions (about 10–20) within a sample are evaluated, and the sample is deemed correct only if all constituent questions yield right result. As shown in Table 2, accuracies of the executable part remain 100% in relaxed evaluation, suggesting the robustness of FoVer. It is also seen that the difference between relaxed and standard evaluation increases, which indicates that the strict evaluation is better at validation and identifying problems of the dataset.

### 5.2 FOLIO Results

Table 3 presents detailed results on the FOLIO dataset. When considering inexecutable samples, accuracies of Logic-LM and FoVer are both lower than those of CoT. However, when considering only executable samples, both outperform CoT.

The executability rate of GPT-4o is unexpectedly low (77.8%). Upon analysis, we find that most execution errors owe to mismatched types (function misapplication mentioned in Section 3.2). Since FOLIO does not distinguish types,[7] we wrote a specific demo for it, appending it to the end of universal demos. This specific demo instructs LLMs to treat all entities as single sort (`Entity`), preventing them from defining on their own. As demonstrated in Table 3, the executability rate significantly improves for both GPT and Claude. While accuracy of executable part decreases, the overall accuracy shows increase.

| Method | | GPT-4o | | | Claude 3.5 Sonnet | | |
|---|---|---|---|---|---|---|---|
| | | Exe | Acc | Total | Exe | Acc | Total |
| CoT | | / | 72.4 | **72.4** | / | 75.4 | 75.4 |
| Logic-LM | | 74.9 | **82.9**$^{*}$ | 62.1$^{*}$ | 64.0 | 85.4$^{*}$ | 54.7$^{*}$ |
| FoVer | −c | 77.8 | 79.7 | 62.1$^{*}$ | 85.2$^{\dagger}$ | 84.4 | 71.9$^{\dagger}$ |
| | +c | 73.4 | 79.9 | 58.6$^{*}$ | 83.3$^{\dagger}$ | **87.0**$^{*}$ | 72.4$^{\dagger}$ |
| FoVer (specific) | −c | **96.1**$^{\dagger}$ | 73.3 | 70.4$^{\dagger}$ | **96.1**$^{\dagger}$ | 74.9$^{\dagger}$ | 71.9$^{\dagger}$ |
| | +c | **96.1**$^{\dagger}$ | 73.8 | 70.9$^{\dagger}$ | 93.6$^{\dagger}$ | 81.1 | **75.9**$^{\dagger}$ |

Table 3: Detailed results on FOLIO (%). ''specific'' denotes results of specific prompt rather than universal prompt. ''+c'' and ''−c'' denote FoVer with and without common knowledge added respectively. ''*'' indicates statistical significance against CoT; ''†'' indicates statistical significance against Logic-LM.

Figure 6 illustrates the confusion matrices for universal and specific prompts when adopting common knowledge. Compared to the universal prompt, specific prompt results in a higher number of samples being classified as Uncertain (U in Figure 6) by the model. This suggests that, under the impact of the additional demo, the executable sample increases, while models' ability to flexibly define predicates as well as their relations is compromised. However, the precision of model output for True and False cases ($P_{\mathrm{TF}}$) remains high, indicating that True and False outputs of FoVer are robustly dependable.

We notice a sample for which both GPT and Claude classify as True under universal prompt, while the label is False (orange box in Figure 6). This sample is classified as Uncertain by GPT-4o when using specific prompt. We conduct a human inspection and find that the conclusion of this sample is an implication, where both the antecedent and consequent are False (shown in Figure 15), which should be evaluated True, suggesting a labeling error in the dataset. While occasional mistakes exist (e.g., the other misclassified case by GPT-4o under universal prompt, shown in Figure 16), the consistent True predictions from both models helps reveal potential issues of the dataset.

[7] For example, the same predicate `isLocatedIn(x, y)` can accept various entities like `isLocatedIn(Book, Library)` or `isLocatedIn(City, Country)` without explicit type specifications.

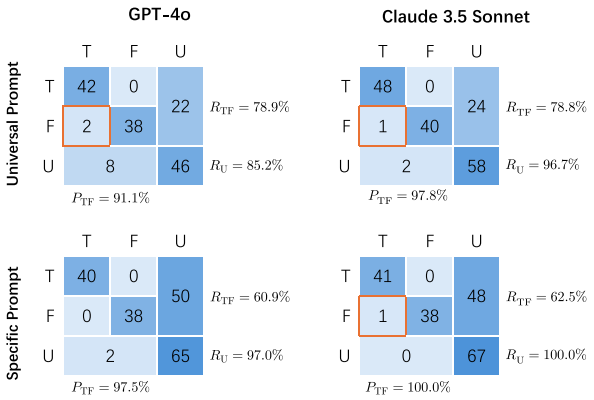



Figure 6: Confusion matrices on FOLIO. Presented here are results with common knowledge added. T represents True, F False, U Uncertain. The ordinate is the ground truth label, and the abscissa is the predicted result. $P_{TF}$ is precision of pipeline outputting True or False, $R_{TF}$ is recall of pipeline outputting True of False, $R_U$ is recall of pipeline outputting Uncertain. Orange box marks exceptional cases detailed in Section 5.2 and Appendix B.

| Method | | GPT-4o | | | Claude 3.5 Sonnet | | |
|---|---|---|---|---|---|---|---|
| | | Exe | Acc | Total | Exe | Acc | Total |
| StrategyQA | | | | | | | |
| Direct | | / | 79.2 | **79.2** | / | 75.8 | **75.8** |
| FoVer | −c | 75.0 | 67.8 | 50.8* | **83.3** | 64.0 | 53.3* |
| | +c | 71.7 | **82.6** | 59.2* | 82.5 | **82.8** | 68.3 |
| MuSiQue | | | | | | | |
| Direct | | / | 55.0 | 55.0 | / | 64.2 | 64.2 |
| FoVer | −c | 80.0 | 87.5* | 70.0* | 91.7 | 78.2 | 71.7 |
| | +c | 80.8 | **91.8*** | **74.2*** | **92.5** | **84.7*** | **78.3*** |

Table 4: Detailed results on REVEAL (%). ''Direct'' denotes the results of single decision. ''*'' indicates statistical significance against CoT.

### 5.3 REVEAL Results

Detailed results on REVEAL are shown in Table 4. On the StrategyQA subset, compared to single decision (labeled as Direct), both LLMs perform better in the executable portions using FoVer, but the overall accuracy is lower due to insufficient executability. On the MuSiQue subset, however, FoVer significantly outperforms single decision.

It is observed that single decision and FoVer behave differently on the two subsets—single decision performs better on StrategyQA, while executability rates of FoVer are lower on this subset. This is attributable to the disparity of the two datasets. As shown in Figure 7, StrategyQA exhibits greater abstraction and ambiguity compared to MuSiQue. The StrategyQA example typ-

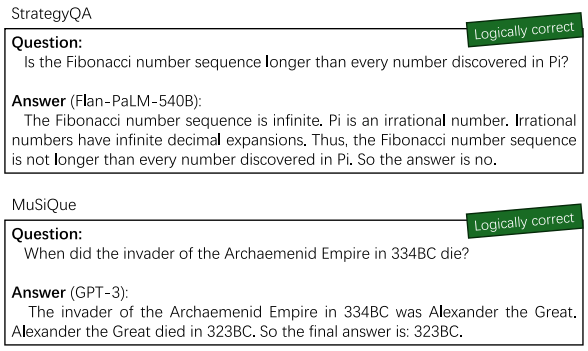



Figure 7: Examples in REVEAL dataset.

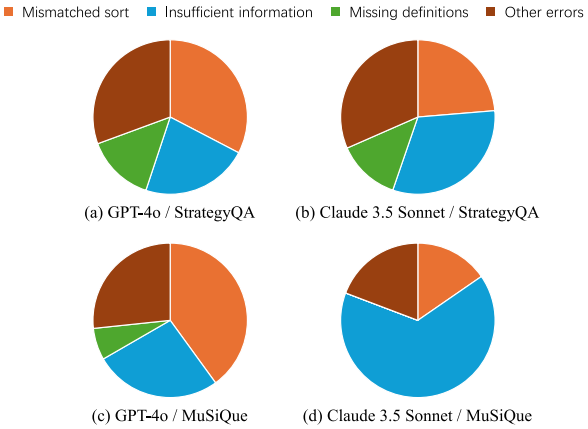



Figure 8: Distribution of primary error types of FoVer (with common knowledge added) on REVEAL.

ically contains linguistically ambiguous phrasing like ''longer than every number discovered in Pi'', which allows multiple interpretations. In contrast, MuSiQue questions are generally grounded in concrete factual information with clear targets, using precise and unambiguous language to ask for specific facts or entity relationships, with straightforward reasoning processes.

A more detailed account of error types is provided in Figure 8. A significant portion of failures can be attributed to the mismatched sorts, which is discussed in Section 3.2. Although the prompt optimization mitigates this issue, it persists, particularly with GPT-4o. Another critical type of error is due to insufficient information, where FoVer outputs Uncertain while the ground truth label is True or False. This error type indicates that, though the integration of common knowledge demonstrates its effectiveness (see Section 5.4), in some instances, it is still insufficient to prove or disprove the conclusion, which suggests a potential future improvement. There is also a smaller fraction of errors stemming from missing definitions,

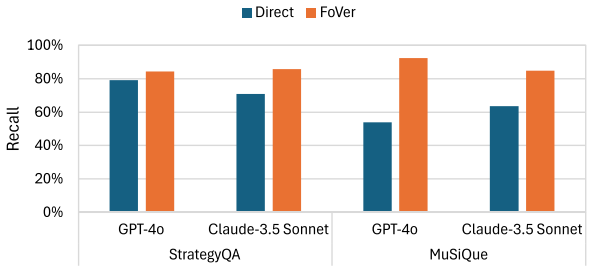



Figure 9: Recall rate of True samples on REVEAL. ''Direct'' denotes the result of single decision.

| Dataset | C | GPT-4o | | | Claude 3.5 Sonnet | | |
|---|---|---|---|---|---|---|---|
| | | Exe | Acc | Total | Exe | Acc | Total |
| ProofWriter | – | 97.0 | 100.0 | 97.0 | 95.5 | 100.0 | 95.5 |
| | + | 91.0* | 100.0 | 91.0* | 92.5* | 100.0 | 92.5* |
| FOLIO | – | 96.1 | 73.3 | 70.4 | 96.1 | 74.9 | 71.9 |
| | + | 96.1 | 73.8 | 70.9 | 93.6 | 81.1* | 75.9 |
| REVEAL-StrategyQA | – | 75.0 | 67.8 | 50.8 | 83.3 | 64.0 | 53.3 |
| | + | 71.7 | 82.6* | 59.2* | 82.5 | 82.8* | 68.3* |
| REVEAL-MuSiQue | – | 80.0 | 87.5 | 70.0 | 91.7 | 78.2 | 71.7 |
| | + | 80.8 | 91.8 | 74.2 | 92.5 | 84.7* | 78.3* |

Table 5: Results with executability rate on all datasets (%). ''C'' stands for common knowledge; ''+'' and ''–'' denote FoVer with and without common knowledge added, respectively. Orange marks decrease compared to results without common knowledge added; green marks increase. ''*'' indicates statistical significance.

where some sorts or entities are used without prior declarations.

Additionally, we conduct an analysis on the improvements of the executable portion of FoVer compared to single decision and find that the increase in accuracy is attributable to improvement in the recall rate of True labels. As shown in Figure 9, recall rates are improved across different models and datasets. In single decision, LLMs often reject correct answers. Sometimes they conflate logical correctness with factuality, but sometimes they are just overconservative.

### 5.4 Impact of Common Knowledge

The impact of common knowledge on FoVer results is summarized in Table 5. Generally, the incorporation of common knowledge brings improvement in accuracy, but has negative impact on executability. The decrease in executability is an inherent consequence of design, as it introduces scenarios where claim expressions are correct but common knowledge expressions contain errors. On ProofWriter, the executability rates

**Premises:**

- All customers in James' family who subscribe to AMC A-List are eligible to watch three movies every week without any additional fees.
- Some of the customers in James' family go to the cinema every week.
- Customers in James' family subscribe to AMC A-List or HBO service.
- Customers in James' family who prefer TV series will not watch TV series in cinemas.
- All customers in James' family who subscribe to HBO services prefer TV series to movies.
- Lily is in James' family; she watches TV series in cinemas.

**Conclusion:** Lily goes to cinemas every week.

**Premises-FOL:**

- $\forall x\,((\text{Customer}(x) \wedge \text{In}(x,\ \text{jameSFamily}) \wedge \text{SubscribedTo}(x,\ \text{aMCAList})) \rightarrow \text{EligibleForThreeFreeMoviesEveryWeekWithoutAdditionalFees}(x))$
- $\exists x \exists y\,(\text{Customer}(x) \wedge \text{In}(x,\ \text{jameSFamily}) \wedge \text{GoToEveryWeek}(x,\ \text{cinema}) \wedge (\neg(x = y)) \wedge \text{Customer}(y) \wedge \text{In}(y,\ \text{jameSFamily}) \wedge \text{GoToEveryWeek}(y,\ \text{cinema}))$
- $\forall x\,(\text{Customer}(x) \wedge \text{In}(x,\ \text{jameSFamily}) \wedge (\text{SubscribedTo}(x,\ \text{aMCAList}) \vee \text{SubscribedTo}(x,\ \text{hBO})))$
- $\forall x\,((\text{Customer}(x) \wedge \text{In}(x,\ \text{jameSFamily}) \wedge \text{Prefer}(x,\ \text{tVSeries})) \rightarrow (\neg\text{WatchIn}(x,\ \text{tV},\ \text{cinema})))$
- $\forall x\,((\text{Customer}(x) \wedge \text{In}(x,\ \text{jameSFamily}) \wedge \text{SubscribedTo}(x,\ \text{hBO})) \rightarrow \text{Prefer}(x,\ \text{tVSeries}))$
- $\text{Customer}(\text{lily}) \wedge \text{In}(\text{lily},\ \text{jameSFamily} \wedge \text{WatchIn}(\text{lily},\ \text{tV},\ \text{cinema})$

**Conslusion-FOL:** $\text{GoToEveryWeek}(\text{lily},\ \text{cinema})$

Figure 10: An example from the FOLIO training set (`example_id` = 1192) illustrating the need for implicit knowledge. Each example in FOLIO contains both natural language and FOL representation of premises and conclusions.

decrease more than on other datasets. This is due to contradiction between translated claims (from given text) and added common knowledge, as the rulebases in ProofWriter are often counter-factual.

On the other hand, on REVEAL-MuSiQue, executability rates are improved on both models after incorporation of common knowledge. This phenomenon is related to another kind of execution error. In a few cases (about 1%), Z3 fails to decide whether given expressions are satisfiable or not, resulting in execution timeout. We surprisingly find that, on MuSiQue, common knowledge helps Z3 to make decisions; some of former undecidable cases become decidable.

The integration of common knowledge improves the accuracy across all configurations, with ProofWriter being the sole exception. This pattern is intriguing because both ProofWriter and FOLIO are specialized logic datasets designed to be self-contained. Theoretically, common knowledge should not contribute to performance on these datasets. While the lack of improvement on ProofWriter aligns with this expectation, the performance gain on FOLIO warrants closer examination.

Our analysis reveals that FOLIO actually benefits from additional implicit premises in certain cases. Consider the example in Figure 10, where the FOL representation includes `Customer (Lily)` while corresponding premise is absent

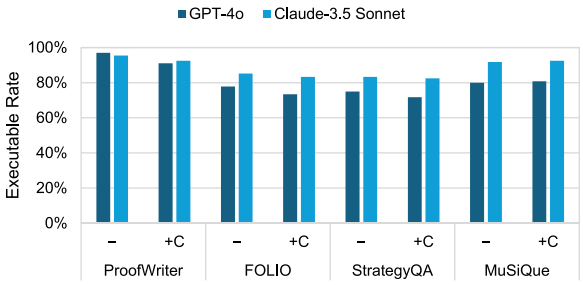



Figure 11: Executability rate of GPT-4o and Claude 3.5 Sonnet on all datasets. ‘‘+C’’ denotes results with common knowledge added.

from the original text. This discrepancy highlights that some FOLIO samples require bridging implications that can be derived from common knowledge. Therefore, the observed performance improvement on FOLIO still aligns with our expectations about the role of common knowledge.

Notably, on the REVEAL dataset, the variation leads to a marked increase in overall accuracy, underscoring the effectiveness of our approach in incorporating knowledge of LLMs.

### 5.5 Comparison between GPT and Claude

We compare the performance of GPT-4o and Claude 3.5 Sonnet. As demonstrated in Table 1, Claude 3.5 Sonnet generally outperforms GPT-4o, on both CoT and FoVer. The performance trends observed with FoVer align closely with those of CoT across different models and datasets, while minor variations exist. This consistency suggests that FoVer could correctly reflect capabilities of its base LLM.

The reason Claude 3.5 Sonnet outperforms GPT-4o with FoVer can be attributed to its higher executability rate. As illustrated in Figure 11, Claude 3.5 Sonnet shows higher executability rates compared to GPT-4o across different benchmarks, with the sole exception of ProofWriter (−) where they show comparable performance. Higher executability correlates to not only better coding abilities avoiding syntax and grammar error, but also higher consistency within response, better understanding of instructions, and greater capabilities of reasoning. These results suggest the potential of FoVer as a method for revealing nuanced differences in language model performance.

### 5.6 Potential Use Cases

Beyond automated logical verification of given text, our method can also be applied to other tasks.

#### 5.6.1 Logical Dataset Verification

As mentioned in Section 5.1, FoVer has identified instances of annotation errors in ProofWriter, which are decidable but labeled as Unknown. It has also found errors and paradoxes in FOLIO.

On both ProofWriter and FOLIO, FoVer demonstrates exceptionally high accuracy, especially for cases outputting True and False, making it robustly dependable. There are many datasets that are constructed automatically following specific rules. Our method could be adopted to verify the validity of data in these datasets. We suggest that future logical dataset construction efforts employ our method for final dataset validation.

#### 5.6.2 First-Order Logic Dataset Construction

To better facilitate LLMs to translate natural language into Z3 expression, we have instructed them to output translations as tuples `(string, Expression)`, where the string element describes the corresponding Z3 expression, as shown in Table 6. This enables us to construct a dataset containing premises and conclusion in natural language as well as corresponding logic label.

## 6 Limitations

We instruct the LLMs to generate entire function code rather than in a statement-by-statement manner, considering their autonomy, enabling them to decide predicates, sorts, and common knowledge systematically and flexibly. While this strategy is effective, and the optimization of prompts has enhanced their ability to generate expressions correctly, LLMs still produce incorrect expressions for individual sentences. In the future, we might employ multi-agent or other methods to automatically check the validity of generated expressions, extending this pipeline to iteratively generate critical expressions.

The proposed method incurs notable costs due to the long length of prompts. However, various policies adopted by LLM providers have already reduced these costs, including Batch API[8,9] and

[8] `https://platform.openai.com/docs/guides/batch`.

[9] `https://docs.anthropic.com/en/docs/build-with-claude/message-batches`.

```
def president_of_us_when_superconductivity_was_discovered_was_woodrowwilson(**kwargs) -> Logic:
    """
    (CoT Annotation ommitted, shown in Figure 3)
    """
    # Initialize an instance of Logic with given arguments.
    l = Logic(**kwargs)

    # Define types.
    Newthing = DeclareSort('Newthing')
    Person = DeclareSort('Person')
    Region = DeclareSort('Region')
    Event = DeclareSort('Event')
    # I shall use these identifiers for placeholders: Newthing: n*, Person: p*, Region: r*, Event: e*; Int: i*, Bool: b*.

    # Define functions with usage comments.
    n_discoverer__person = Function('discoverer', Newthing, Person) # (Newthing) -> Person, usage: n_discoverer__person(Newthing) = Person.
    p_is_president_of_r_in_i = Function('is-president-of-in', Person, Region, IntSort(), BoolSort()) # (Person, Region, Int) -> Bool, usage: p_is_president_of_r_in_i(Preson, Region, Int). # Person is president of Region from Int a to Int b means Person is president of Region in Int x (a <= x <= b).
    president_of_r_when_e_happen__person = Function('president-of-when', Region, Event, Person) # (Region, Event) -> Person, usage: president_of_r_when_e_happen__person(Region, Event) = Person.
    discover_n_as__event = Function('discover', Newthing, Event) # (Newthing) -> Event, usage: discover_n_as__event(Newthing) = Event.
    e_happentime__int = Function('happentime', Event, IntSort()) # (Event) -> Int, usage: e_happentime__int(Event) = Int.

    # Arrange instances.
    superconductivity = Const('superconductivity', Newthing)
    heikekamerlinghonnes = Const('Heike Kamerlingh Onnes', Person)
    woodrowwilson = Const('Woodrow Wilson', Person)
    unitedstates = Const('United States', Region)
    us = Const('U.S.', Region)

    # I'm not sure what quantifiers will be used, so I shall define them later.
    def _store():
        # Relation Definitions
        l.definitions = [
            # What does president of Region when Event happen is Person mean.
            (
                "President of Region when an Event happened was Person if and only if the Event happened in the year that Person was president of that Region.",
                ForAll([p1, e1, r1], (president_of_r_when_e_happen__person(r1, e1) == p1) == Exists([i1], And(e_happentime__int(e1) == i1, p_is_president_of_r_in_i(p1, r1, i1))))
            ),
            # Necessary constraints for 1-to-1 relations.
            (
                "'discover New thing is Event' is an injective relation between New thing and Event.",
                ForAll([n1, n2, e1], Implies(And(discover_n_as__event(n1) == e1, discover_n_as__event(n2) == e1), n1 == n2))
            )
        ]
        # Claims from text
        l.claims = [
            ("Superconductivity was discovered in 1911.", e_happentime__int(discover_n_as__event(superconductivity)) == 1911),
            (
                "Superconductivity was discovered by Heike Kamerlingh Onnes.",
                n_discoverer__person(superconductivity) == heikekamerlinghonnes
            ),
            (
                "Woodrow Wilson was president from 1913 to 1921.",
                ForAll([i1], Implies(And(i1 >= 1913, i1 <= 1921), (p_is_president_of_r_in_i(woodrowwilson, unitedstates, i1))))
            ),
        ]
        # Common sense
        l.common_knowledge = [
            ("U.S. is United States.", us == unitedstates),
        ]
        # Target.
        l.assertions = [(
            "President of the U.S. when superconductivity was discovered was Woodrow Wilson.",
            president_of_r_when_e_happen__person(us, discover_n_as__event(superconductivity)) == woodrowwilson
        )]

    # All placeholders used: n1, n2: Newthing, p1: Person, e1: Event, r1: Region, i1: Int
    n1, n2 = Consts('n1 n2', Newthing)
    p1, = Consts('p1', Person)
    e1, = Consts('e1', Event)
    r1, = Consts('r1', Region)
    i1, = Ints('i1')

    _store()

    return l
```

Table 6: The first demo within the FoVer few-shot prompt. The Question-Answer pair is shown in Figure 1a; the CoT annotation is shown in Figure 3. Some comments are omitted to save space. Full prompt is available in our GitHub repository.[1]

Prompt Caching.[10,11] With the reduction trends of API prices, the cost of our method is expected to decrease over time.

## 7 Conclusions

In this work, we propose FoVer, an automated pipeline to verify the logical correctness of reasoning texts. It effectively incorporates world knowledge that LLMs gained during pre-training, contributing to their performance. Unlike previous methods that are restricted to specific formats, FoVer achieves strong generality by leveraging LLMs' natural language understanding capabilities while maintaining rigorous logical verification through theorem provers. The experiments demonstrate that FoVer outperforms existed methods in logical verification significantly, with prominent improvement of reliability.

As two of its potential applications, FoVer can be employed to validate existing logical reasoning datasets and to automatically construct new datasets from natural language inference texts, further contributing to the advancement of research in this field.

Our method is extensible. Although current work focuses on logical validation, this pipeline is well-suited for fact verification. In the future, we may draw upon document retrieval methods, utilizing relevant knowledge base information for fact verification.

## Acknowledgments

We thank Zong and Lin (2024) and Press et al. (2023) for their inspiring works, which directly motivates this study. We also sincerely thank the editor and reviewers for their valuable comments and suggestions, which have helped us improve this work significantly. Finally, this work is supported by the National Key R&D Program of China under grant no. 2023YFB3308004 and National Natural Science Foundation under grant no. 92267107.

## References

Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. 2022. Cvc5: A versatile and industrial-strength SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Cham. Springer International Publishing. https://doi.org/10.1007/978-3-030-99524-9_24

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Meiqi Chen, Yubo Ma, Kaitao Song, Yixin Cao, Yan Zhang, and Dongsheng Li. 2024. Improving large language models in event relation logical prediction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9451–9478, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.512

Donald Davidson. 1966. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press. https://doi.org/10.2307/jj.13027259.6

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer. https://doi.org/10.1007/978-3-540-78800-3_24

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu,

[10] https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching.

[11] https://platform.openai.com/docs/guides/prompt-caching.

Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948v1*. https://doi.org/10.48550/arXiv.2501.12948

Bruce Frederiksen. 2008. Applying expert system technology to code reuse with Pyke. In *PyCon*. Chicago.

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. https://doi.org/10.1162/tacl_a_00370

Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2022. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024a. FOLIO: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840v3*. https://doi.org/10.48550/arXiv.2209.00840

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024b. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608v7*. https://doi.org/10.48550/arXiv.2403.14608

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong

as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.254

Ashwin Kalyan, Abhinav Kumar, Arjun Chandrasekaran, Ashish Sabharwal, and Peter Clark. 2021. How much coffee was consumed during EMNLP 2019? Fermi problems: A new reasoning challenge for AI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7318–7328, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.582

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*, volume 35, pages 22199–22213.

Abhinav Lalwani, Lovish Chopra, Christopher Hahn, Caroline Trippel, Zhijing Jin, and Mrinmaya Sachan. 2024. NL2FOL: Translating natural language to first-order logic for logical fallacy detection. *arXiv preprint arXiv:2405.02318v1*. https://doi.org/10.48550/arXiv.2405.02318

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.229

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. LogiQA 2.0—An improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2947–2962. https://doi.org/10.1109/TASLP.2023.3293046

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 *Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 4, pages 3622–3628. https://doi.org/10.24963/ijcai.2020/501

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.ijcnlp-main.20

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.173

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157. https://doi.org/10.1007/BF02295996, PubMed: 20254758

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435v10*. https://doi.org/10.48550/arXiv.2307.06435

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. https://doi.org/10.18653/v1/2023.emnlp-main.313

OpenAI. 2024. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin

Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 technical report. *arXiv preprint arXiv:2303.08774v6*. https://doi.org/10.48550/arXiv.2303.08774

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1504–1518, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.129

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*, pages 3806–3824, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.248

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.378

Ori Press, Andreas Hochlehnert, Ameya Prabhu, Vishaal Udandarao, Ofir Press, and Matthias Bethge. 2024. CiteME: Can Language Models Accurately Cite Scientific Claims? In *Advances in Neural Information Processing Systems*, volume 37, pages 7847–7877. Curran Associates, Inc.

Qwen Team. 2024. QwQ: Reflect deeply on the boundaries of the unknown. http://qwenlm.github.io/blog/qwq-32b-preview/

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii,

Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis, Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T., Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W. Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala

Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

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*, pages 3621–3634, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.317

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1421

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554. https://doi.org/10.1162/tacl_a_00475

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6942–6959, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.375

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*, volume 36, pages 45548–45580.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models. *arXiv preprint arXiv:2303.18223v15*. https://doi.org/10.48550/arXiv.2303.18223

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Shi Zong and Jimmy Lin. 2024. Categorical syllogisms revisited: A review of the logical reasoning abilities of LLMs for analyzing categorical syllogism. *arXiv preprint arXiv:2406.18762v1*. https://doi.org/10.48550/arXiv.2406.18762, https://doi.org/10.18653/v1/2024.nlp4science-1.20

## A Few-Shot Demos

Demos for CoT is shown in Figure 12. The prompts for ProofWriter and FOLIO are adapted from Pan et al. (2023), converting the text-completion format into chat-based format suitable for the latest LLMs, with added XML tags to structure assistant demonstrations.

ProofWriter 2-shot CoT

**System**
Given a problem statement as contexts, the task is to decide whether the target is True, False, or Unknown.

**User**
Context:
The cow is blue. (More context here···) If the cow is cold and the cow visits the lion then the lion sees the squirrel.

Target: The tiger is not young.

**Assistant**
<thinking>The tiger likes the cow. (More thinking here···) So the tiger is young.</thinking>
<answer>False</answer>

(The second User/Assistant pair···)

FOLIO 2-shot CoT

**System**
Given a problem statement as contexts, the task is to decide whether the target is True, False, or Uncertain.

**User**
Context:
The Blake McFall Company Building is a commercial warehouse listed on the National Register of Historic Places. (More context here···) John works at the Emmet Building.

Target: The Blake McFall Company Building is located in Portland, Oregon.

**Assistant**
<thinking>The tiger likes the cow. (More thinking here···) So the tiger is young.</thinking>
<answer>False</answer>

(The second User/Assistant pair···)

REVEAL single decision

**System**
Given a Question and an Answer, the task is to decide whether the Answer is logically correct, regardless of factuality.
You may think first, but must give one of [true, false] at last.

Figure 12: Prompt example for CoT and single decision.

Table 6 presents a prompt demo of FoVer with CoT annotation omitted. The Question-Answer pair of this example is shown in Figure 1a; the CoT annotation is shown in Figure 3. The rest of the demos are available in our GitHub repository.

## B Exceptional Cases

Figure 13 illustrates a ProofWriter example of decidable question which is incorrectly labeled Unknown; Figure 14 shows a paradox theory. These errors indicate that the construction of ProofWriter considers only forward rules, excluding contrapositives of implications.

FOLIO also contains the wrong label for conclusion. Figure 15 shows an example where the conclusion is an implication with False antecedent and False consequent, but the label is False.

Figure 16 presents the misclassified case by GPT-4o on FOLIO (orange box in Figure 6).

.....
Fiona is quiet. Fiona is not smart. Fiona is white.
.....
If someone is big then they are cold.
.....
All quiet, blue people are big.
Cold, quiet people are smart.
.....

**Target:** Fiona is **not blue**.

Figure 13: Example of decidable question in the ProofWriter development set. Green marks positive conclusions; orange marks negative conclusions.

.....
Fiona is not green. Fiona is kind.
.....
Blue people are quiet.
.....
If someone is **round** then they are green.
Quiet people are round.
If someone is kind and not blue then they are **round**.

Figure 14: Example of paradox theory in the ProofWriter development set. Green marks positive conclusions; orange marks negative conclusions; red marks the paradox.

1. If a restaurant is listed in Yelp's recommendations, then the restaurant has not received many negative reviews.
2. All restaurants with a rating greater than four are listed in Yelp's recommendations.
3. Some restaurants that do not provide take-out service receive many negative reviews.
4. All restaurants that are popular among local residents have ratings greater than four.
5. The Hamden Plaza Subway store has a rating greater than four, or it is popular among local residents.

**Conclusion:**
If the Hamden Plaza Subway store provides take-out service and receives many negative reviews, then its rating is greater than 4 and it does not provide take-out service.

Figure 15: Example in the FOLIO validation set that is True but labeled False.

GPT-4o

```
l.assertions = [(
  "If Yuri is not an American professional basketball player, then Yuri is a
  professional basketball player.",
  Implies(And(p__nationality(yuri) != american, p__profession(yuri) ==
  professional_basketball_player), p__profession(yuri) ==
  professional_basketball_player)
)]
```

Claude 3.5 Sonnet

```
l.assertions = [(
  "If Yuri is not an American professional basketball player, then Yuri is a
  professional basketball player.",
  Implies(
    Not(And(p_is_american_national(yuri), p_is_pro_basketball_player(yuri))),
    p_is_pro_basketball_player(yuri)
  )
)]
```

Figure 16: Error example of GPT-4o on FOLIO. Orange highlights the wrong translation. In contrast, Claude 3.5 Sonnet translates correctly.

Sometimes LLMs make mistakes in individual cases. Future work may introduce iterative generation and inspection, reducing such errors.