LOGIC-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning

Liangming Pan **Alon Albalak** **Xinyi Wang** **William Yang Wang**

University of California, Santa Barbara

{liangmingpan, alon_albalak, xinyi_wang, wangwilliamyang}@ucsb.edu

Abstract

Large Language Models (LLMs) have shown human-like reasoning abilities but still struggle with complex logical problems. This paper introduces a novel framework, LOGIC-LM, which integrates LLMs with symbolic reasoning to improve logical problem-solving. Our method first utilizes LLMs to translate a natural language problem into a symbolic formulation. Afterward, a deterministic symbolic solver performs inference on the formulated problem. We also introduce a self-refinement stage, which utilizes the symbolic solver's error messages to revise symbolic formalizations. We demonstrate LOGIC-LM's effectiveness on four logical reasoning datasets: ProofWriter, PrOntoQA, FOLIO, and LogicalDeduction. Our results show significant improvement compared to LLMs alone, with an average performance boost of 62.6% over standard prompting and 23.5% over chain-of-thought prompting. Our findings suggest that LOGIC-LM, by combining LLMs with symbolic logic, offers a promising avenue for faithful logical reasoning.[1]

1 Introduction

Logical reasoning is a cognitive process that involves using evidence, arguments, and logic to arrive at conclusions or make judgments (Huang and Chang, 2022). It plays a central role in intelligent systems for problem-solving, decision-making, and critical thinking. Recently, large language models (LLMs) (Brown et al., 2020; Ouyang et al., 2022a; OpenAI, 2023) have exhibited emergent ability to "reason" like human (Wei et al., 2022a). When prompted with step-wise explanations of reasoning ("chain of thoughts"), or a simple prompt "Let's think step by step.", these models are able to answer questions with explicit reasoning steps (Wei et al., 2022b; Kojima et al., 2022).

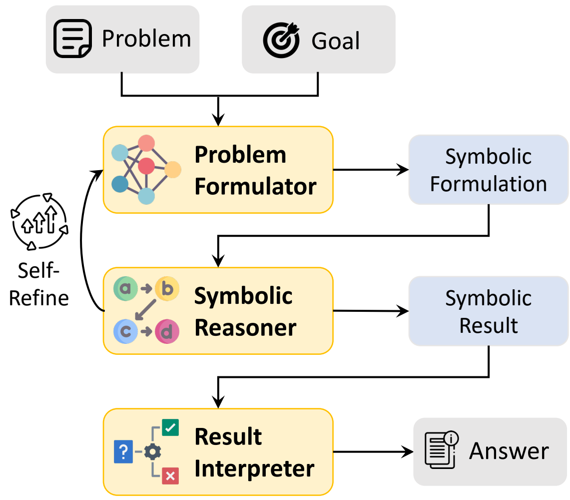

Figure 1: Overview of our LOGIC-LM framework.

Despite the advances of LLMs, they still struggle with complex logical reasoning problems (Liu et al., 2023b). Recent studies (Golovneva et al., 2023; Ribeiro et al., 2023; Lyu et al., 2023) found that LLMs occasionally make *unfaithful* reasoning, *i.e.*, the derived conclusion does not follow the previously generated reasoning chain. While chain-of-thought may imitate human reasoning processes, the fundamental nature of LLMs remains that of black-box probabilistic models, lacking a mechanism to guarantee the faithfulness of reasoning (Shanahan, 2022). In contrast, *symbolic inference engines*, such as expert systems (Metaxiotis et al., 2002), are faithful and transparent because the reasoning is based on symbolic-represented knowledge and follows well-defined inference rules that adhere to logical principles. The main obstacle is how to accurately translate a problem into symbolic representations, considering the inherent ambiguity and flexibility of natural language. This is precisely where LLMs excel, making LLMs a promising complement to symbolic solvers.

This complementarity motivates our exploration of neuro-symbolic methods that integrate language models with symbolic reasoning. As illustrated in

[1]Code and data are publicly available at https://github.com/teacherpeterpan/Logic-LLM.

Figure 1, we present LOGIC-LM, a novel framework that decomposes a logical reasoning problem into three stages: *Problem Formulation*, *Symbolic Reasoning*, and *Result Interpretation*. During the problem formulation stage, an LLM converts a natural language description of the problem into an appropriate symbolic formulation, identifying key entities, facts, and rules present in the problem statement. Subsequently, at the symbolic reasoning stage, a deterministic symbolic solver performs inference on the symbolic formulation. Lastly, a result interpreter explains the output and maps it to the correct answer. By incorporating LLMs with symbolic solvers, we can exploit the robust natural language understanding capabilities of LLMs to precisely represent the problem using symbolic representations, while also taking advantage of the logical faithfulness and transparency offered by symbolic inference engines. To improve the accuracy of the symbolic parsing, we also incorporate the idea of self-refinement to iteratively revise the generated logical form using the error messages from the symbolic solver as feedback.

We showcase the adaptability and effectiveness of LOGIC-LM on four logical reasoning datasets: *ProofWriter* (Tafjord et al., 2021), *PrOntoQA* (Saparov and He, 2023), *FOLIO* (Han et al., 2022), and the *LogicalDeduction* dataset from BigBench (Srivastava et al., 2022). These datasets cover a wide range of logical reasoning problems, including deductive reasoning, first-order logic (FOL) reasoning, and constraint satisfaction problems (CSP). We integrate three types of symbolic inference tools tailored to these problems: 1) *logic programming* engine that supports deductive reasoning through forward/backward chaining; 2) *FOL inference engine* that derives new conclusions based on FOL rules and facts, and 3) *constraint optimization* engine that provides solvers for CSP over finite domains. Our evaluations show that the strategy of integrating LLMs with symbolic solvers performs significantly better than purely relying on LLMs for logical reasoning, with an average improvement of 62.6% over the standard prompting and 23.5% over the chain-of-thought prompting (§ 4.1). We also find that LOGIC-LM becomes increasingly effective as the required reasoning depth increases (§ 4.2). Finally, by analyzing the impact of self-refinement, we highlight the effectiveness of incrementally revising symbolic formalizations when interacting with the symbolic solver (§ 4.3).

2 Related Work

Language Models for Logical Reasoning. Recent works in adapting large language models for logical reasoning tasks can be broadly categorized into two groups: 1) *fine-tuning approaches* that optimize LLMs' reasoning ability through fine-tuning or training specialized modules (Clark et al., 2020; Tafjord et al., 2022; Yang et al., 2022), and 2) *in-context learning approaches* that design special prompts to elicit LLMs' step-by-step reasoning capabilities. Typical methods include chain-of-thought prompting (Wei et al., 2022b; Wang et al., 2022b) that generates explanations before the final answer and the least-to-most prompting (Zhou et al., 2022) that breaks the problem down into simpler components that can be solved individually. Both the above approaches perform reasoning directly over natural language (NL), providing greater flexibility than symbolic-based reasoning. However, the intrinsic complexity and ambiguity of NL also bring undesired issues such as unfaithful reasoning and hallucinations.

Different from prior works, we use *symbolic language* as the basic unit of reasoning. This effectively transfers the burden of executing complex, precise reasoning from LLMs to more reliable, interpretable external symbolic solvers. Simultaneously, we leverage the strong in-context learning ability of LLMs to formulate the NL-based problem into suitable symbolic representations, thus maintaining the benefit of flexibility.

Although prior works (Mao et al., 2019; Gupta et al., 2020; Manhaeve et al., 2021; Cai et al., 2021; Tian et al., 2022; Pryor et al., 2022) also propose neuro-symbolic methods to combine neural networks with symbolic reasoning, these methods suffer from limitations such as hand-crafted or specialized module designs that are not easily generalizable, or brittleness due to the difficulty of optimization. In contrast, we propose a more generalizable framework that integrates modern LLMs with symbolic logic without the need for training or designing complex problem-specific modules.

Tool-augmented Language Models. Language models have inherent limitations such as the inability to access up-to-date information, take actions, or perform precise mathematical reasoning. To address this, recent work has begun to augment language models with access to external tools and resources, such as the information re-

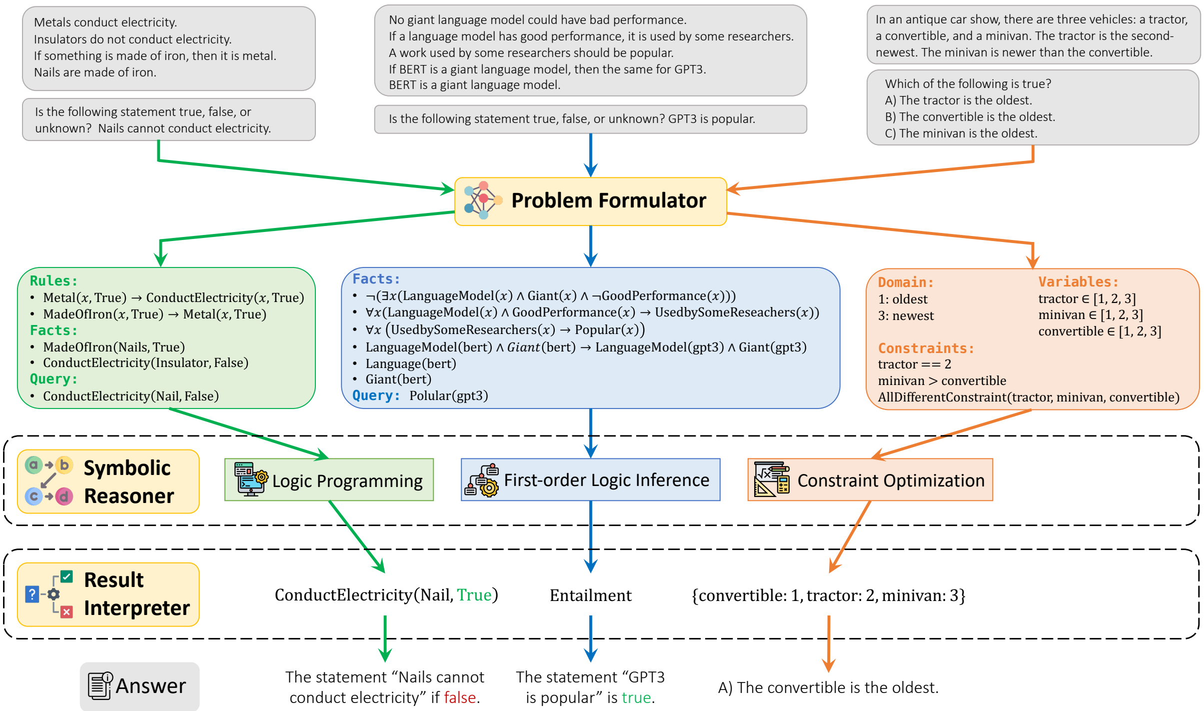

Figure 2: Overview of our LOGIC-LM model, which consists of three modules: (*1*) *Problem Formulator* generates a symbolic representation for the input problem with LLMs via in-context learning (*2*) *Symbolic Reasoner* performs logical inference on the formulated problem, and (*3*) *Result Interpreter* interprets the symbolic answer.

triever (Nakano et al., 2021; Shi et al., 2023; Lazaridou et al., 2022), calculator (Cobbe et al., 2021), code interpreter (Wang et al., 2022a), planner (Liu et al., 2023a), and other pre-trained models (Shen et al., 2023). For mathematical reasoning, several works have shown that the integration of a calculator (Cobbe et al., 2021; Imani et al., 2023) or a Python interpreter (Gao et al., 2022; Chen et al., 2022) into LLMs can greatly enhance performance by offloading numerical calculations. However, this idea has not been extended to general logic reasoning problems, primarily due to the challenge of representing their reasoning procedure with mathematical expressions or Python programs. Our work provides a novel way to handle complex logical reasoning within the framework of augmented LLMs, by using symbolic language and offloading the reasoning to external symbolic solvers.

3 LOGIC-LM

As shown in Figure 2, the inputs of our model are a logical reasoning problem P described in natural language, along with a goal G in the form of a multiple-choice or free-form question. LOGIC-LM then follows a *problem formulation-and-reasoning* paradigm to solve the problem.

In the *Problem Formulation* stage, we prompt an LLM to translate the problem and the goal into a task-specific symbolic language. In the *Symbolic Reasoning* stage, we call a deterministic symbolic solver, *e.g.*, a logic programming engine, to obtain a symbolic-represented answer. Finally, an LLM- or rule-based *Result Interpreter* is responsible for translating the answer back to natural language. Using this approach, the reasoning is guaranteed to be faithful as long as the problem formulation is correct since the answer A is the result of executing deterministic algorithms (*e.g.*, forward/backward-chaining) embedded within symbolic reasoners. Compared to previous methods based on chain-of-thought, our framework reduces the burden on LLMs by shifting their focus from “reasoning step-by-step to solve the problem” to “representing the problem in symbolic language”.

3.1 Problem Formulator

Intuitively, LLMs may struggle with directly solving complex reasoning problems. However, they have demonstrated a notable ability to comprehend textual inputs and translate them into formal programs, such as mathematical equations (He-Yueya et al., 2023) or Python codes (Gao et al., 2022). We posit that this capability to *formulate problems into different languages* can be extended to symbolic languages as well. We leverage the few-shot generalization ability of LLMs to achieve this. By providing the LLMs with detailed instructions about the grammar of the symbolic language, alongside a few demonstrations as in-context examples, we observe that LLMs, like InstructGPT (Ouyang et al., 2022b) and GPT-4 (OpenAI, 2023), can effectively follow the instructions to identify key entities, facts, and rules present in the problem statement, and then translate these elements into symbolic language following our defined grammar.

3.2 Symbolic Language Grammar

Specifically, we design three sets of grammar corresponding to three common types of logical reasoning problems: *deductive reasoning*, *first-order logic reasoning*, and *constraint satisfaction*. These grammars provide a foundation for translating problem statements into symbolic languages. By creating additional problem-specific grammars, our framework retains the flexibility to accommodate a wider range of reasoning tasks. Next, we will delve into each type of symbolic grammar. Examples of each problem type are in Figure 2.

Logic Programming for Deductive Reasoning. Deductive reasoning typically starts from known facts and rules, and iteratively makes new inferences until the goal statement can be proved or disproved (Poole and Mackworth, 2010). The `Prolog` logic programming language (Clocksin and Mellish, 2003; Körner et al., 2022) is arguably the most prominent symbolic language to describe deductive reasoning problems. We adopt its grammar to represent a problem as facts, rules, and queries.

- **Facts**: a fact F is a simple statement with a *predicate* and a set of *arguments*, formulated as $P(a_1, \cdots, a_n)$, where P is the predicate name and each argument a_i can be a variable, entity, number, or bool. For example, `Age(Peter, 31)` means "Peter's age is 31", and `MadeOfIron(Nails, True)` represents the fact "Nails are made of iron".
- **Rules**: rules are written in the form of clauses: $F_1 \wedge \cdots \wedge F_m \rightarrow F_{m+1} \wedge \cdots \wedge F_n$, where each F_i is a fact and the rule means "if the facts $F_1, \cdots, F_m$ are true, then the facts $F_{m+1} \cdots F_n$ are also true."
- **Queries**: a query Q is simply another fact required to be proved based on known facts and rules.

First-Order Logic Reasoning. While the logic programming language efficiently represents common deductive reasoning problems, it may fail to represent more complex first-order logic (FOL) problems. To address this, we also include the FOL grammar (Enderton, 2001) in Appendix A. A problem is then parsed into a list of FOL formulas, which are divided into *Facts* (the known information from the problem) and *Query* (the unknown formula to be proved). To facilitate the inference, we employ the textual notations defined in the Stanford CS221 course, which mirror the mathematical notations. An example is given in Table 1.

Constraint Satisfaction. Another common type of logical reasoning is constraint satisfaction problems (CSPs) (Kumar, 1992), defined as a set of objects whose state must satisfy a number of constraints or limitations. A CSP is often defined as a triple (X, D, C), where $X = \{x_1, \cdots, x_n\}$ is a set of variables, $D = \{D_1, \cdots, D_n\}$ is a set of their respective domains of values, and $C = \{C_1, \cdots, C_m\}$ is a set of constraints. Each variable X_i can take on the values in the nonempty domain D_i. Every constraint C_j is a pair $\langle t_j, R_j \rangle$, where $t_j \subset X$ is a subset of k variables and R_j is a k-ary relation on the corresponding subset of domains D_j. We use the above syntax as the symbolic representation for a CSP problem, consisting of variables, domains, and constraints. An example is given in both Figure 2 and Table 1.

Prompts. Appendix B shows the full prompts we use for the problem formulator. To teach LLMs to better align each statement with its corresponding symbolic form, we use the format SYMBOLIC_FORMULA ::: NL_STATEMENT in in-context examples to enable better grounding.

3.3 Symbolic Reasoner

After the problem formulator parses the problem P and the goal G into symbolic representations $\hat{P}$ and $\hat{G}$, we call a deterministic external solver, *e.g.*, an expert system, a FOL prover, or a CSP solver, depending on the task, to obtain the answer A.

Expert System. For deductive reasoning, we incorporate the Pyke expert system (Frederiksen, 2008), which makes inferences based on the logic programming language. In response to a query, `Pyke` first creates a knowledge base, populating it with known facts and rules. Subsequently, it applies forward- and backward-chaining algorithms to infer new facts and substantiate the goal.

Problem	Statement	Symbolic Representation	
		Mathematical Notation	Textual Notation
Deductive Reasoning	If the circuit is complete and the circuit has the light bulb then the light bulb is glowing.	$\texttt{Complete(Circuit, True)} \wedge$ $\texttt{Has(Circuit, LightBulb)}$ $\rightarrow \texttt{Glowing(LightBulb, True)}$	Complete(Circuit, True) && Has(Circuit, LightBulb) >>> Glowing(LightBulb, True)
First-Order Logic	A Czech person wrote a book in 1946.	$\exists x_2 \exists x_1 (\texttt{Czech}(x_1) \wedge \texttt{Author}(x_2, x_1)$ $\wedge \texttt{Book}(x_2) \wedge \texttt{Publish}(x_2, 1946))$	Exists($x2, Exists($x1, AndList([Atom('Czech', $x1), Atom('Author', $x2, $x1), Atom('Book', $x2), Atom('Publish', $x2, 'year1946')])))
Constraint Satisfaction	On a shelf, there are five books. The blue book is to the right of the yellow book.	$\texttt{blue\_book} \in \{1,2,3,4,5\}$ $\texttt{yellow\_book} \in \{1,2,3,4,5\}$ $\texttt{blue\_book} > \texttt{yellow\_book}$	blue_book [IN] [1,2,3,4,5] yellow_book [IN] [1,2,3,4,5] blue_book > yellow_book

Table 1: Examples of the symbolic language we use for three different types of logical reasoning problems.

FOL Prover. We use the FOL inference engine from the one provided on the Stanford CS221 course page[2], following Han et al. (2022). The inference engine initially converts FOL statements to conjunctive normal form (CNF) and then performs resolution (Robinson, 1965) on the CNF to deduce whether a conclusion is true, false, or unknown.

CSP Solver. Solving a CSP is to find value assignments for all variables that satisfy all given constraints. Commonly used algorithms for this task include backtracking, constraint propagation, and local search variants. To this end, we incorporate the `python-constraint`[3] package which offers solvers for CSPs over finite domains.

3.4 Self-Refiner

For complex problems, generating the correct logical form may become challenging for LLMs. To address this, we introduce a *self-refinement* module that learns to modify inaccurate logical representations using the error messages from the symbolic reasoner as feedback. Recent works (Chen et al., 2023; Madaan et al., 2023) have adopted similar ideas to improve code generation, by teaching LLMs to debug their predicted programs via few-shot demonstrations. Here we extend this idea to refine generated logic representations. If the symbolic solver returns an execution error, we instruct the LLM to refine the incorrect logical form, by prompting it with the erroneous logic form, the solver's error message, and a set of demonstrations showing common error cases and their remedies, *e.g.*, a free variable is not bounded to any quantifier in FOL. We run this process iteratively until either no error messages are returned, or the maximum number of allowable iterations is reached.

Dataset	Reasoning	Test Size	#Opts
PrOntoQA	Deductive	500	2
ProofWriter	Deductive	600	3
FOLIO	FOL	204	3
LogicalDeduction	CSP	300	3,5,7

Table 2: Statistics of the logical reasoning datasets.

3.5 Result Interpreter

Finally, the result interpreter translates the results returned from the symbolic solver back to a natural language answer. For certain problems, this can be achieved through predefined rules; for example, mapping `Entailment` to `true`. However, the process can be more complex for CSPs, *e.g.*, translating *{convertible: 1, tractor: 2, minivan: 3}* to "*the convertible is the oldest.*". To handle these varying levels of complexity, we designed both rule-based and LLM-based result interpreters.

4 Experiments

Datasets. We evaluate LOGIC-LM on four common logical reasoning datasets, as follows.

PrOntoQA (Saparov and He, 2023) is a recent synthetic dataset created to analyze the capacity of LLMs for deductive reasoning. We use the hardest *fictional characters* version of the dataset, based on the results in Saparov and He (2023). Each version is divided into different subsets depending on the number of reasoning hops required. We use the hardest 5-hop subset for evaluation. Each question in PrOntoQA aims to validate a new fact's veracity, such as "True or false: Alex is not shy.".

ProofWriter (Tafjord et al., 2021) is another commonly used dataset for deductive logical reasoning. Compared with PrOntoQA, the problems are expressed in a more naturalistic language form.

[2]https://stanford-cs221.github.io/spring2022/assignments/logic/index.html

[3]https://github.com/python-constraint/python-constraint

Dataset	GPT-3.5 (text-davinci-003)			GPT-4 (gpt-4)		
	Standard	CoT	Logic-LM	Standard	CoT	Logic-LM
PrOntoQA	51.80	91.00	**93.20**	77.40	**98.79**	93.60
ProofWriter	36.16	48.33	**70.11**	52.67	68.11	**79.33**
FOLIO	54.60	57.84	**61.76**	69.11	70.58	**74.50**
LogicalDeduction	41.33	48.33	**67.66**	71.33	75.25	**89.29**

Table 3: Accuracy of standard promoting (Standard), chain-of-thought promoting (CoT), and our method (LOGIC-LM) on four reasoning datasets. The best results within each base LLM (GPT-3.5 and GPT-4) are highlighted.

We use the open-world assumption (OWA) subset in which each example is a (problem, goal) pair and the label is one of $\{$*PROVED*, *DISPROVED*, *UNKNOWN*$\}$. The dataset is divided into five parts, each part requiring 0, ≤ 1, ≤ 2, ≤ 3, and ≤ 5 hops of reasoning, respectively. We evaluate the hardest *depth-5* subset. To reduce overall experimentation costs, we randomly sample 600 examples in the test set and ensure a balanced label distribution.

FOLIO (Han et al., 2022) is a challenging expert-written dataset for logical reasoning. The problems are mostly aligned with real-world knowledge and use highly natural wordings, and the questions require complex first-order logic reasoning to solve. We use the entire FOLIO test set for evaluation, consisting of 204 examples.

LogicalDeduction is a challenging logical reasoning task from the BigBench (Srivastava et al., 2022) collaborative benchmark. The problems are mostly about deducing the order of a sequence of objects from a minimal set of conditions. We use the full test set consisting of 300 examples.

Table 2 gives the data statistics. We convert all examples into a standardized multiple-choice format, comprising a problem statement, a question, and potential answers, as shown in Figure 2. We also select five examples from the training set of each dataset as in-context examples.

Baselines. We compare our model against two baselines that depend solely on LLMs for logical reasoning: 1) *Standard* LLMs, which leverage in-context learning to directly answer the question; and 2) *Chain-of-Thought* (CoT) (Wei et al., 2022b), which adopts a step-by-step problem-solving approach, generating explanations before providing the final answer. We separately evaluate the settings that GPT-3.5 (`text-davinci-003`) (Ouyang et al., 2022a) and GPT-4 (`gpt-4`) (OpenAI, 2023) serve as the underlying LLMs for all models. To ensure fair comparisons, we use $N = 2$ in-context examples for all models. For reproducible results, we set the temperature to 0 and select the response with the highest probability from LLMs. Since all examples are formed as multiple-choice questions, we evaluate model performance based on the accuracy of selecting the correct answer.

4.1 Main Results

We report the results of LOGIC-LM and baselines in Table 3. We have three major observations.

First, Logic-LM significantly outperforms standard LLMs and CoT across all datasets, with GPT-3.5 and GPT-4 serving as base LLMs. With GPT-3.5, our method outperforms standard LLM on all datasets, with an average improvement of 62.6%. This highlights the benefit of combining LLMs with external symbolic solvers for logical reasoning. LOGIC-LM also improves CoT by a large margin of 23.5% on average, showing that symbolic reasoning with LOGIC-LM is more faithful than the NL-based reasoning with CoT.

Second, the standard LLM performance exhibits a significant average increase of 48.56% when switching from GPT-3.5 to GPT-4. This observation aligns with the GPT-4 technical report's assertion that the main enhancement of GPT-4 lies in its ability to carry out complex reasoning (OpenAI, 2023). Although this may indicate that the logical reasoning capability can be boosted by scaling up the LLM, we observe that GPT-4 still makes numerous unfaithful reasoning errors. By delegating the reasoning to symbolic solvers, our method can further improve GPT-4's logical reasoning performance by an average of 26.12%.

Thirdly, while integrating CoT generally enhances LLM performance, we find its benefits comparatively less substantial on FOL and CSP problems with a modest improvement of 7.61% versus a 41.56% enhancement on deductive reasoning

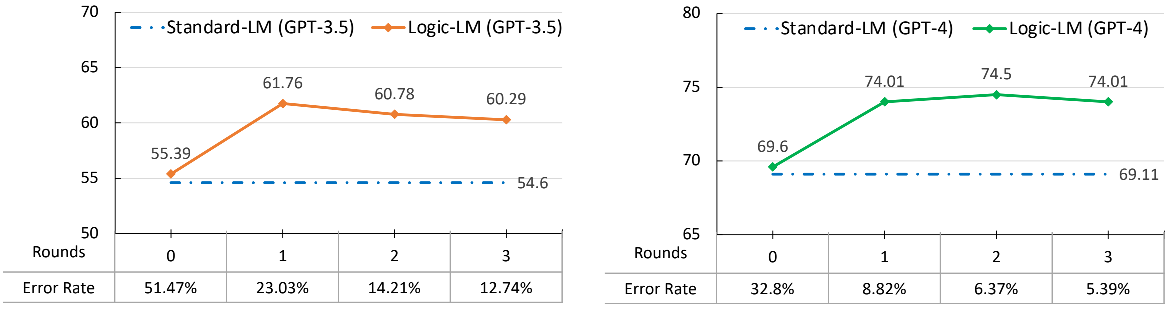

Figure 3: Ablation study: the accuracy for different rounds of self-refinement, with the corresponding error rates.

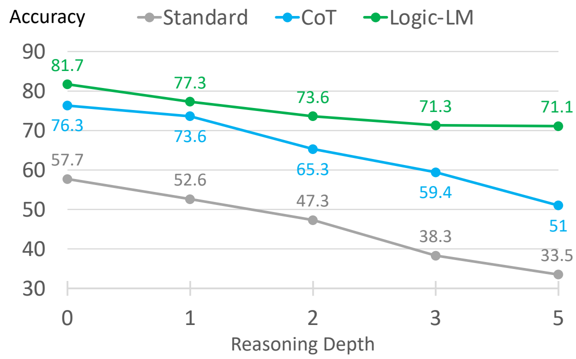

Figure 4: Accuracy of different models for increasing size of reasoning depth on the ProofWriter dataset.

problems. A plausible explanation is that CoT emulates human forward-chain reasoning: beginning with known facts and sequentially deriving new conclusions until the goal is met. This reasoning style aligns well with problems in the PrOntoQA and ProofWriter datasets. However, FOL and CSP problems often necessitate more sophisticated reasoning strategies that are "non-linear" compared to standard forward-chain reasoning. These include hypothesizing, case-by-case analysis, and the process of elimination. Compared to CoT, the integration of symbolic solvers is better suited to these reasoning styles, hence yielding a more marked improvement on the FOLIO (+10.5%) and the LogicalDeduction (+44.4%) datasets.

4.2 How Does the Symbolic Solver Help?

To better understand how symbolic solvers facilitate logical reasoning, we evaluate the performance of LOGIC-LM and baselines for questions of varying complexity levels. To this end, we randomly selected 300 examples from each subset of ProofWriter, ensuring a balanced label distribution. The problems in these subsets require 0, <=1, <=2, <=3, and <=5 hops of reasoning, respectively. The results, shown in Figure 4, indicate that LOGIC-LM becomes increasingly effective as the required reasoning depth increases. For example, LOGIC-LM outperforms CoT by 7.1%, 5.0%, 12.7%, 20.0%, and 39.4% on depth-0, depth-1, depth-2, depth-4, and depth-5 problems, respectively. In LOGIC-LM, multi-step logical reasoning is delegated to external symbolic solvers, thereby transitioning the challenge of LLM from *problem-solving* to *problem representation*. Ideally, the complexity of formally representing a problem statement in logical form should remain relatively constant, regardless of whether the questions require simple or complex reasoning. The trends in Figure 4 validate this assumption. The performance of *Standard* and *CoT* declines precipitously with the escalation of problem complexity. However, this trend is less prominent for LOGIC-LM, indicating that the robust reasoning capabilities provided by external solvers substantially mitigate performance degradation for complex reasoning problems.

4.3 Impact of Self-Refinement

We then explore the effects of self-refinement by performing an ablation study on FOLIO. We evaluate performance across different rounds of self-refinement, namely, 0 (no refinement), 1, 2, and 3 rounds. We also record the corresponding 'error rate' for each round, defined as the percentage of invalid logical representations resulting in execution errors from the symbolic solver. The results are depicted in Figure 3.

Our observations reveal that without any self-refinement, the initial error rates exceed 50% and 30% when employing GPT-3.5 and GPT-4 as problem formulators respectively, highlighting the challenge of accurately translating a problem statement into a valid logical format. Nevertheless, as the rounds of self-refinement increase, the error rate consistently declines, leading to a significant enhancement in the final performance. This suggests that self-refinement serves as an effective tool in

```
Problem: There are six types of wild turkeys: Eastern wild turkey, Osceola wild turkey, Gould's wild turkey, Merriam's wild turkey, Rio Grande
wild turkey, and Ocellated wild turkey. Tom is not an Eastern wild turkey. Tom is not an Osceola wild turkey. Tom is also not a Gould's wild
turkey, or a Merriam's wild turkey, or a Rio Grande wild turkey. Tom is a wild turkey.
Question: Based on the above information, is the following statement true, false, or uncertain? Tom is an Ocellated wild turkey.
(A) True              (B) False            (C) Uncertain

Predicted logic programs:
    Facts:
    AndList([Atom('WildTurkey', 'eastern'), Atom('WildTurkey', 'osceola'), Atom('WildTurkey', 'goulds'), Atom('WildTurkey', 'merriams'),
    Atom('WildTurkey', 'rio_grande'), Atom('WildTurkey', 'ocellated')])
    → Forall($x1, Implies(Atom('WildTurkey', '$x1'), OrList([Atom('OcellatedWildTurkey', '$x1'), Atom('EasternWildTurkey', '$x1'),
    Atom('OsceolaWildTurkey', '$x1'), Atom('GouldsWildTurkey', '$x1'), Atom('MerriamsWildTurkey', '$x1'), Atom('RioGrandeWildTurkey', '$x1')])))
    AndList([Not(Atom('EasternWildTurkey', 'tom')), Not(Atom('OsceolaWildTurkey', 'tom')), Not(Atom('GouldsWildTurkey', 'tom')),
    Not(Atom('MerriamsWildTurkey', 'tom')), Not(Atom('RioGrandeWildTurkey', 'tom')), Atom('WildTurkey', 'tom')])
    Query:
    Atom('OcellatedWildTurkey', 'tom')
Predicted answer: C → A
```

```
Problem: A Japanese game company created the game the Legend of Zelda. All games in the Top 10 list are made by Japanese game companies. If a
game sells more than one million copies, then it will be selected into the Top 10 list. The Legend of Zelda sold more than one million copies.
Question: Based on the above information, is the following statement true, false, or uncertain? The Legend of Zelda is in the Top 10 list.
(A) True              (B) False            (C) Uncertain

Predicted logic programs:
    Facts:
    Atom('JapaneseGameCompany', 'japanese_game_company')
    Forall('$x1', Implies(Atom('Top10List', '$x1'), Atom('JapaneseGameCompany', '$x1')))
    Forall('$x1', Implies(Atom('SellMoreThanOneMillionCopies', '$x1'), Atom('Top10List', '$x1')))
    Atom('SellMoreThanOneMillionCopies', 'legend_of_zelda')
    Query:
    Atom('Top10List', 'legend_of_zelda')
Predicted answer: A
```

```
Problem: The following paragraphs each describe a set of five objects arranged in a fixed order. The statements are logically consistent within
each paragraph. In a golf tournament, there were five golfers: Rob, Eve, Eli, Amy, and Dan. Dan finished second. Amy finished below Eve. Dan
finished above Eve. Amy finished above Eli.
Question: Which of the following is true?
(A) Rob finished third   (B) Eve finished third   (C) Eli finished third   (D) Amy finished third   (E) Dan finished third

Predicted logic programs:
    Domain:                Variables:                  Constraints:
    1: first               Rob [IN] [1, 2, 3, 4, 5]    Dan == 2
    5: last                Eve [IN] [1, 2, 3, 4, 5]    Amy > Eve
                           Eli [IN] [1, 2, 3, 4, 5]    Dan > Eve → Dan < Eve
                           Amy [IN] [1, 2, 3, 4, 5]    Amy < Eli
                           Dan [IN] [1, 2, 3, 4, 5]    AllDifferentConstraint([Rob, Eve, Eli, Amy, Dan])
    Query:
    (A) Rob == 3    (B) Eve == 3    (C) Eli == 3    (D) Amy == 3    (E) Dan == 3
Predicted answer: E → B
```

Figure 5: Examples of generated symbolic representations and predicted answers. The incorrect segment(s) and their correspondence in the problem are marked in yellow, and the correct revisions are marked in green.

aiding the LLM to accurately frame the problem.

Despite the substantial improvement achieved with a single round of self-refinement, we note that performance stagnates in subsequent rounds, even though the error rate continues to drop. This can be attributed to the type of feedback received by the self-refiner, which is the error message from the symbolic solver. This feedback aids in converting "invalid" symbolic representations into valid ones. However, a valid symbolic representation does not necessarily equate to a "correct" problem formulation that accurately represents the problem. This issue could be tackled by enhancing the self-refiner to incorporate feedback beyond the error message, *e.g.*, a reward signal from an additional module evaluating the accuracy of a generated symbolic form. We leave this for future exploration.

4.4 Case Study

In Figure 5, we present three typical examples of symbolic representations generated by GPT-4, together with the predicted answers by symbolic solvers. In general, GPT-4 has demonstrated a potent capacity to interpret complex problems into symbolic forms, as exemplified by the middle example. Nonetheless, there remain certain difficulties with regard to accurately understanding the semantics of the problem.

The top example illustrates an instance where GPT-4 generates an incorrect FOL representation, stemming from its inability to define appropriate predicates. Here, instead of creating the predicate `EasternWildTurkey`, the model generates a constant, `WildTurkey(eastern)`, in which `WildTurkey` is the predicate and `eastern` is the constant. While this representation is valid in isolation, it does not interact well with subsequent constants, where the predicate `EasternWildTurkey` is defined again. This inconsistency is a recurring issue in GPT-4's symbolic form generation, illustrating that the model sometimes struggles to maintain an overarching understanding of the problem when forming logical symbols. The bottom example highlights a case where GPT-4 struggles to interpret specific expressions accurately. In this case, the model fails to distinguish between the meanings of "below" and "above", resulting in an incorrect constraint `Dan > Eve`. These error cases underscore that transforming problems into logical forms remains a challenging task for GPT-4, due to the innate flexibility of natural language and the complexity of global problem comprehension.

5 Conclusion and Future Work

In this work we propose a novel approach to address logical reasoning problems by combining large language models with symbolic solvers. We introduce Logic-LM, one instantiation of such a framework, and demonstrate how it significantly improves performance over pure LLMs and chain-of-thought prompting techniques.

While Logic-LM has proven to be a capable system, it can be further improved with extension to more flexible and powerful logic systems. For example, statistical relational learning (SRL) systems (e.g. Markov logic networks (Richardson and Domingos, 2006) or probabilistic soft logic (Bach et al., 2017)) have demonstrated great promise in reasoning under uncertainty and integration with our framework would enable even more adaptive problem solving capabilities. Additionally, our method can be extended to reasoning problems requiring commonsense, which remain a significant challenge as they often require reasoning over complex and ambiguous rules.

References

Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 18(1):1–67.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Le-Wen Cai, Wang-Zhou Dai, Yu-Xuan Huang, Yu-Feng Li, Stephen H. Muggleton, and Yuan Jiang. 2021. Abductive learning with ground knowledge base. In *The International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1815–1821. ijcai.org.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *CoRR*, abs/2211.12588.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *CoRR*, abs/2304.05128.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *The International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3882–3890.

William F Clocksin and Christopher S Mellish. 2003. *Programming in PROLOG*. Springer Science & Business Media.

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

Herbert B Enderton. 2001. *A mathematical introduction to logic*. Elsevier.

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. 2022. PAL: program-aided language models. *CoRR*, abs/2211.10435.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *International Conference on Learning Representations (ICLR)*.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *International Conference on Learning Representations (ICLR)*.

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.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*.

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

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *CoRR*, abs/2303.05398.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916.

Philipp Körner, Michael Leuschel, João Barbosa, Vítor Santos Costa, Verónica Dahl, Manuel V. Hermenegildo, José F. Morales, Jan Wielemaker, Daniel Diaz, and Salvador Abreu. 2022. Fifty years of prolog and beyond. *Theory Pract. Log. Program.*, 22(6):776–858.

Vipin Kumar. 1992. Algorithms for constraint-satisfaction problems: A survey. *AI Mag.*, 13(1):32–44.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *CoRR*, abs/2203.05115.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. LLM+P: empowering large language models with optimal planning proficiency. *CoRR*, abs/2304.11477.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023b. Evaluating the logical reasoning ability of chatgpt and GPT-4. *CoRR*, abs/2304.03439.

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.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2021. Neural probabilistic logic programming in deepproblog. *The Journal of Artificial Intelligence (AIJ)*, 298:103504.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations (ICLR)*.

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

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

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022b. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

David Poole and Alan K. Mackworth. 2010. *Artificial Intelligence - Foundations of Computational Agents*. Cambridge University Press.

Connor Pryor, Charles Dickens, Eriq Augustine, Alon Albalak, William Wang, and Lise Getoor. 2022. Neupsl: Neural probabilistic soft logic.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henry Zhu, Rui Dong, Deguang Kong, Juliette Burger, Anjelica Ramos, William Yang Wang, Zhiheng Huang, George Karypis, Bing Xiang, and Dan Roth. 2023. STREET: A multi-task structured reasoning and explanation benchmark. In *International Conference on Learning Representations (ICLR)*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1):107–136.

John Alan Robinson. 1965. A machine-oriented logic based on the resolution principle. *The Journal of the ACM (JACM)*, 12(1):23–41.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations (ICLR)*.

Murray Shanahan. 2022. Talking about large language models. *CoRR*, abs/2212.03551.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in huggingface. *CoRR*, abs/2303.17580.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652.

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, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew K. 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 Karakas, and et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *CoRR*, abs/2206.04615.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3621–3634.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2078–2093.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2022. Weakly supervised neural symbolic learning for cognitive tasks. In *The AAAI Conference on Artificial Intelligence (AAAI)*, pages 5888–5896.

Xingyao Wang, Sha Li, and Heng Ji. 2022a. Code4struct: Code generation for few-shot structured prediction from natural language. *CoRR*, abs/2210.12810.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 89–105. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *CoRR*, abs/2205.10625.

A Syntax for First-order Logic (FOL)

Name	Math Notation	Textual Notation
Constant	lowercase letters	lowercase letters
Variable	$x_1, x_2, \cdots$	\$x1, \$x2, $\cdots$
Atom	$P(a_1, \cdots, a_n)$	`Atom('P', a1, ⋯, an)`
Negation	$\neg P$	`Not(P)`
Conjunction	$P_1 \wedge P_2$ $P_1 \wedge, \cdots, \wedge P_n$	`And(P1, P2)` `AndList(P1, ⋯, Pn)`
Disjunction	$P_1 \vee P_2$ $P_1 \vee, \cdots, \vee P_n$	`Or(P1, P2)` `OrList(P1, ⋯, Pn)`
Implication	$P_1 \rightarrow P_2$	`Implies(P1, P2)`
Equivalence	$P_1 \leftrightarrow P_2$	`Equiv(P1, P2)`
Existential Quantifier	$\exists x P(x, \cdots)$	`Exists($x, Atom('P', $x, ⋯))`
Universal Quantifier	$\forall x P(x, \cdots)$	`Forall($x, Atom('P', $x, ⋯))`

Table 4: First-Order Logic Grammar.

B Prompt Examples

In this section we provide examples of the prompts used for each dataset and method. Prompts for standard in-context learning contain 2 demonstrations consisting of 3 parts each: a context, a question, and options. Prompts for chain-of-thought prompting contain 2 demonstrations consisting of 5 parts each: a task description, a context, a question, options, and a chain of reasoning. Prompts for Logic-LM contain 2 demonstrations with 5 parts each: a task description, a context, a question, options, and a domain specific symbolic program. For brevity we show only a single demonstration in the following sections.

B.1 PrOntoQA Prompts

Standard In-Context Learning

```
Context: Jompuses are not shy. Jompuses are yumpuses.
(··· more context here ···)
Zumpuses are rompuses. Max is a yumpus.

Question: Is the following statement true or false?
Max is sour.

Options:
A) True
B) False

The correct option is:
```

Chain-of-Thought Prompting

```
Task Description: Given a problem statement as
contexts, the task is to answer a logical reasoning
question.

Context: Jompuses are not shy. Jompuses are yumpuses.
(··· more context here ···)
Zumpuses are rompuses. Max is a yumpus.

Question: Is the following statement true or false?
Max is sour.

Options:
A) True
B) False

Reasoning: Max is a yumpus. Each yumpus is a dumpus.
(··· more reasoning here ···)
Tumpuses are not sour. So Max is not sour.

The correct option is:
```

Logic-LM

```
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

Context: Each jompus is fruity.
(··· more context here ···)
Rompuses are zumpuses. Alex is a tumpus.

Question: True or false: Alex is not shy.

Predicates:
Jompus(\$x, bool) ::: Does x belong to Jompus?
(··· more predicates here ···)
Zumpus(\$x, bool) ::: Does x belong to Zumpus?

Facts:
Tumpuses(Alex, True)

Rules:
Jompus($x, True) >>> Fruity($x, True)
(··· more rules here ···)
Dumpus(\$x, True) >>> Rompus(\$x, True)

Query:
Shy(Alex, False)
```

B.2 ProofWriter Prompts

Standard In-Context Learning

```
Context: The cow is blue. The cow is round.
(··· more context here ···)
If the cow is cold and the cow visits the lion then
the lion sees the squirrel.

Question: Based on the above information, is the
following statement true, false, or unknown?
The tiger is not young.

Options:
A) True
B) False
C) Unknown

The correct option is:
```

Chain-of-Thought Prompting

```
Task Description: Given a problem statement as
contexts, the task is to answer a logical reasoning
question.

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

Question: Based on the above information, is the
following statement true, false, or unknown?
The tiger is not young.

Options:
A) True
B) False
C) Unknown

Reasoning: The tiger likes the cow.
The tiger likes the squirrel.
(··· more reasoning here ···)
If something is nice and it sees the tiger then
it is young. So the tiger is young.

The correct option is:
```

Logic-LM

```
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

Context: Anne is quiet. Erin is furry.
(··· more context here ···)
All red people are young.

Question: Based on the above information, is the
following statement true, false, or unknown?
Anne is white.

Predicates:
Quiet($x, bool) ::: Is x quiet?
Furry($x, bool) ::: Is x furry?
(··· more predicates here ···)
White($x, bool) ::: Is x white?
Young($x, bool) ::: Is x young?

Facts:
Quite(Anne, True) ::: Anne is quiet.
(··· more facts here ···)
White(Harry, True) ::: Harry is white.

Rules:
Young($x, True) >>> Furry($x, True) ::: Young people
     are furry.
(··· more rules here ···)
Red($x, True) >>> Young($x, True) ::: All red people
     are young.

Query:
White(Anne, True) ::: Anne is white
```

B.3 FOLIO Prompts

Standard In-Context Learning

```
Context: All people who regularly drink coffee are
dependent on caffeine.
(··· more context here ···)
If Rina is not a person dependent on caffeine and
a student, then Rina is either a person dependent
on caffeine and a student, or neither a person
dependent on caffeine nor a student.

Question: Based on the above information, is the
following statement true, false, or uncertain? Rina
is a person who jokes about being addicted to
caffeine or unaware that caffeine is a drug.

Options:
A) True
B) False
C) Uncertain

The correct option is:
```

Chain-of-Thought Prompting

```
Task Description: Given a problem statement as
contexts, the task is to answer a logical reasoning
question.

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.

Question: Based on the above information, is the
following statement true, false, or uncertain?
The Blake McFall Company Building is located in
Portland, Oregon.

Options:
A) True
B) False
C) Uncertain

Reasoning: The Blake McFall Company Building is
another name for the Emmet Building.
(··· more reasoning here ···)
Therefore, the Blake McFall Company Building is
located in Portland, Oregon.

The correct option is:
```

Logic-LM

```
Task Description: You are given a problem description
and a question. The task is to parse the problem
into first-order logic facts and rules and parse
the question into the logic form. We define the
following logical operations:
1) And(arg1, arg2) - logical conjunction of arg1 and
     arg2
2) Or(arg1, arg2) - logical disjunction of arg1 and
     arg2
3) Implies(arg1, arg2) - logical implication of arg1
     and arg2
4) Not(arg1) - logical negation of arg1
5) Xor(arg1, arg2) - logical exclusive disjunction
     of arg1 and arg2
6) Forall(variable, arg1) - logical universal
     quantification of arg1 with respect to variable
7) Exists(variable, arg1) - logical existential
     quantification of arg1 with respect to variable
8) Atom(predicate, arg1, arg2, ...) - logical atom
     with predicate and arguments
9) AndList([arg1, arg2, ...]) - logical conjunction
     of a list of arguments
10) OrList([arg1, arg2, ...]) - logical disjunction
     of a list of arguments
Output format for each fact: logic form :::
     description

Context: All people who regularly drink coffee are
dependent on caffeine.
(··· more context here ···)
If Rina is not a person dependent on caffeine and a
student, then Rina is either a person dependent
on caffeine and a student, or neither a person
dependent on caffeine nor a student.

Question: Based on the above information, is the
following statement true, false, or uncertain?
Rina is either a person who jokes about being
addicted to caffeine or is unaware that caffeine
is a drug.

Facts:
Forall('$x1', Implies(Atom('RegularlyDrinkCoffee' '
     $x1'), Atom('DependentOnCaffeine', '$x1'))) :::
      All people who regularly drink coffee are
     dependent on caffeine.
(··· more facts here ···)
Forall('$x1', Xor(Atom('RegularlyDrinkCoffee', '$x1'
     ), Atom('JokeAboutBeingAddictedToCaffeine', '
     $x1'))) ::: People either regularly drink
     coffee or joke about being addicted to caffeine
     .

Query:
Xor(Atom('JokeAboutBeingAddictedToCaffeine', 'rina')
     , Atom('UnawareThatCaffeineIsADrug', 'rina'))
     ::: Rina is either a person who jokes about
     being addicted to caffeine or is unaware that
     caffeine is a drug.
```

B.4 LogicalDeduction Prompts

Standard In-Context Learning

```
Context: The following paragraphs each describe a
set of seven objects arranged in a fixed order.
(··· more context here ···)
Eve finished below Ada. Rob finished below Joe.

Question: Which of the following is true?

Options:
A) Ana finished third.
B) Eve finished third.
C) Ada finished third.
D) Dan finished third.
E) Rob finished third.
F) Amy finished third.
G) Joe finished third.

The correct option is:
```

Chain-of-Thought Prompting

```
Task Description: Given a problem statement as
contexts, the task is to answer a logical reasoning
question.

Context: The following paragraphs each describe a
set of five objects arranged in a fixed order.
(··· more context here ···)
The raven is the third from the left.

Question: Which of the following is true?

Options:
A) The quail is the rightmost.
B) The owl is the rightmost.
C) The raven is the rightmost.
D) The falcon is the rightmost.
E) The robin is the rightmost.

Reasoning: The owl is the leftmost. This means owl
is not the rightmost.
(··· more reasoning here ···)
This means raven is also not the rightmost. So,
the answer is: A) The quail is the rightmost.

The correct option is:
```

Logic-LM

```
Task Description: You are given a problem description.
The task is to parse the problem as a constraint
satisfaction problem, defining the domain,
variables, and contraints.

Context: The following paragraphs each describe a
set of three objects arranged in a fixed order.
(··· more context here ···)
The minivan is newer than the convertible.

Question: Which of the following is true?

Options:
A) The station wagon is the second-newest.
B) The convertible is the second-newest.
C) The minivan is the second-newest.

Domain:
1: oldest
3: newest

Variables:
station\_wagon [IN] [1, 2, 3]
convertible [IN] [1, 2, 3]
minivan [IN] [1, 2, 3]

Constraints:
station\_wagon == 1 ::: The station wagon is the
    oldest.
minivan > convertible ::: The minivan is newer than
    the convertible.
AllDifferentConstraint([station\_wagon, convertible,
    minivan]) ::: All vehicles have different
    values.

Query:
A) station\_wagon == 2 ::: The station wagon is the
    second-newest.
B) convertible == 2 ::: The convertible is the
    second-newest.
C) minivan == 2 ::: The minivan is the second-newest
    .
```