Language Models can be Deductive Solvers

Jiazhan Feng[1*] **Ruochen Xu**[2] **Junheng Hao**[2] **Hiteshi Sharma**[2]
Yelong Shen[2] **Dongyan Zhao**[1,3] **Weizhu Chen**[2]
[1]Wangxuan Institute of Computer Technology, Peking University, Beijing
[2]Microsoft AI, Redmond
[3]National Key Laboratory of General Artificial Intelligence, Beijing
{fengjiazhan,zhaody}@pku.edu.cn
{ruox,junhenghao,hitshar,yeshe,wzchen}@microsoft.com

Abstract

Logical reasoning is a fundamental aspect of human intelligence and a key component of tasks like problem-solving and decision-making. Recent advancements have enabled Large Language Models (LLMs) to potentially exhibit reasoning capabilities, but complex logical reasoning remains a challenge. The state-of-the-art, solver-augmented language models, use LLMs to parse natural language logical questions into symbolic representations first and then adopt external logical solvers to take in the symbolic representations and output the answers. Despite their impressive performance, any parsing errors will inevitably result in the failure of the execution of external logical solvers and no answer to the logical questions. In this paper, we introduce LoGiPT, a novel language model that directly internalizes and emulates the reasoning processes of logical solvers and avoids parsing errors by learning strict adherence to solver syntax and grammar. LoGiPT is fine-tuned on a newly constructed instruction-tuning dataset derived from revealing and refining the invisible reasoning process of deductive solvers. Experimental results on two public deductive reasoning benchmarks show that LoGiPT outperforms state-of-the-art solver-augmented LMs and few-shot prompting methods on competitive LLMs like GPT-4. This project is available in https://github.com/Cyril-JZ/LoGiPT.

1 Introduction

Logical reasoning is a foundational element of human intelligence, holding a pivotal role in tasks like problem-solving, decision-making, and critical thinking (Huang and Chang, 2023). Recently, substantial advancements have been achieved in the field of NLP through the development of large language models (LLMs) (OpenAI, 2022, 2023; Google, 2023; Touvron et al., 2023a,b). It has been noted that language models (LMs) could potentially display reasoning capabilities when they reach a certain scale threshold (e.g., training compute, model parameters, etc.) (Kaplan et al., 2020; Wei et al., 2022a; Hoffmann et al., 2022). To this end, LLMs can answer logical questions with explicit reasoning steps when prompted with a simple snippet: "*Let's think step by step.*" (Kojima et al., 2022) or step-wise explanations of reasoning (i.e., "chain of thoughts") (Wei et al., 2022b).

While LLMs have made significant progress, complex logical reasoning remains challenging (Valmeekam et al., 2022; Liu et al., 2023b; Luo et al., 2023). Some prior works (Tafjord et al., 2022; Ling et al., 2023) aimed to enable LMs to perform logical reasoning via specialized module fine-tuning, where reasoning is in natural language (NL). However, the ambiguity and complexity of NL can lead to undesired issues such as hallucinations and unfaithful reasoning (Saparov and He, 2023; Gao et al., 2023). To this end, recent work has begun to augment LLMs with access to external **Solvers** (Chen et al., 2022; Ye et al., 2023; Pan et al., 2023). In this paper, we focus on the logical solvers, which are theorem provers that can be any automated reasoning tool for checking the truth value of logical formulas in symbolic language (SL). Invoking logical solvers can guarantee the accuracy of logical reasoning and relieve the burden of LLMs to execute intricate and precise deductive reasoning.

The workflow of the aforementioned solver-augmented LMs is depicted in Figure 1(a). At the outset, the information on the logical questions is stored in NL. It is subsequently fed into a LM for parsing into a symbolic representation suitable for solver-input format. Finally, the SL information is dispatched to a symbolic solver, which yields the truth value of the logical question. However, during this process, any NL-to-SL parsing

[*] Work done during Jiazhan's internship at Microsoft AI, Redmond. Correspondence to: Ruochen Xu and Dongyan Zhao.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 4026–4042
June 16-21, 2024 ©2024 Association for Computational Linguistics

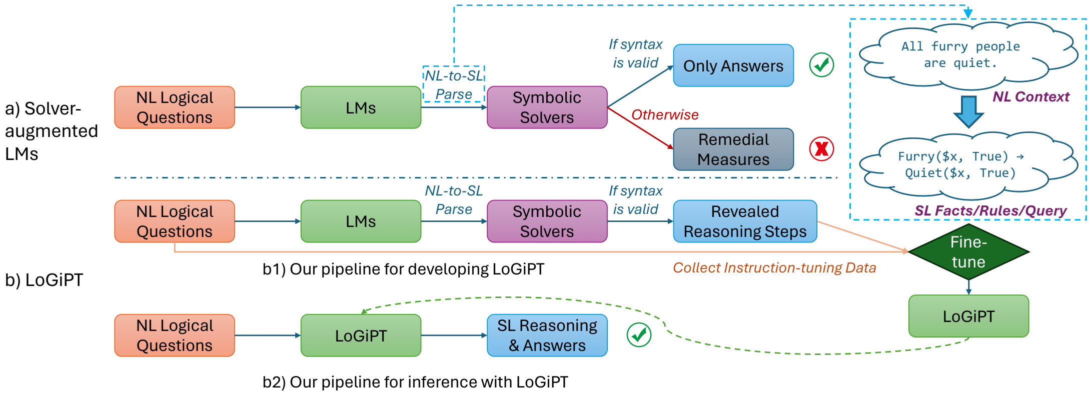

Figure 1: Workflow of current solver-augmented LMs **(a)**, and our pipeline for LoGiPT **(b)**. It is important to note that the NL-to-SL parsing is solely utilized for constructing instruction-tuning data but not for inference process.

errors will inevitably result in the failure of the reasoning process and no answer to the question. In our preliminary experiments (refer to Table 1 in §2.3), we observe that the parsing successful rate (i.e., percentage of executable logical formulations) of Vicuna-13B (Chiang et al., 2023) on ProofWriter (Tafjord et al., 2021) is only 17%. Current methods for solving NL-to-SL parsing failures have two main approaches: directly using LLMs to reason in NL, or regenerating parsing results based on the solver's erroneous message. However, these methods do not solve the problem fundamentally.

In this paper, we introduce LoGiPT, a novel **Logi**c-enhanced **P**re-trained **T**ransformer designed to mimic the reasoning process of logical solvers, enabling it to solve deductive reasoning tasks. We first construct an instruction-tuning dataset containing NL logical questions and their corresponding solver's symbolic reasoning process. After filtering out cases having invalid syntax, we fine-tune open-source LMs like Vicuna or CodeLlama (Roziere et al., 2023) with this data to create LoGiPT. Then, LoGiPT can generate all implied facts given premises and rules, allowing us to determine the truth value of a logical query by matching it with implied facts or outputting "unknown" if it cannot be determined. Our pipeline is presented in Figure 1(b). We can avoid the syntax or grammatical errors derived from NL-to-SL parsing by directly outputting answers with LoGiPT.

Our approach is akin to the process of distillation (Hinton et al., 2015; Xu et al., 2024), whereby we distill knowledge from a symbolic model (i.e., solver) into a neural network (i.e., LM). However, the reasoning process of solvers is invisible to users and we can **only** obtain the answers without intermediate reasoning steps. We design a pipeline to reveal and formalize solvers' invisible reasoning processes, creating instruction-tuning datasets with visible and interpretable symbolic reasoning steps (see Figure 3).

Our main contributions are three-fold:

- To the best of our knowledge, we are the first to propose empowering LLMs to directly learn the reasoning process of logical solvers, thereby acquiring similar reasoning capability for addressing deductive reasoning tasks.
- Our proposed LoGiPT, can directly act as a deductive solver and output all facts implied from NL logical questions while bypassing the syntax or grammatical errors derived from NL-to-SL parsing of solver-augmented LMs.
- Evaluation results on two public deductive reasoning datasets show that LoGiPT can outperform state-of-the-art solver-augmented LMs, and few-shot prompting methods on competitive LLMs like GPT-4.

2 Preliminary

2.1 Deductive Reasoning

Deductive reasoning is an essential type of logical reasoning problem. It typically commences with known facts and rules from logical context, then proceeds through a series of inference steps until the query can be proved or disproved (Poole and Mackworth, 2010). We also adopt the `Prolog` (Clocksin and Mellish, 2003) language, which stands as the most prominent SL for describing deductive reasoning problems. We showcased a deductive reasoning question along with its `Prolog` syntax representation in Figure 2.

For each question, we denote the NL description as **Context**. The **Context** can further be parsed

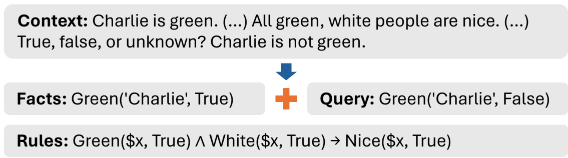

Figure 2: An example of deductive reasoning question from ProofWriter and its parsed Facts, Rules, and Query.

into **Facts**, **Rules**, and **Query**[1]. Specifically, a **Fact** $F = P(a_1, \cdots, a_t)$ is a symbolic statement with a predicate P and t arguments $\{a_1, \cdots, a_t\}$ where a_i can be a variable, entity, number or bool. For example, `Green('Charlie', True)` means "Charlie is green"; **Rules** are presented in the form of clauses $F_1 \wedge \cdots \wedge F_m \rightarrow F_{m+1} \wedge \cdots \wedge F_n$, where F_i is a Fact. The Rule means "if each $F_i \in \{F_1, \cdots, F_m\}$ is true, then we can imply that all Facts in $\{F_{m+1}, \cdots, F_n\}$ are also true." For example, `Furry($x, True) → Quiet($x, True)` indicates if variable `$x` is furry, then `$x` is quiet; a **Query** Q is also in the format of a Fact that needs to be proved based on Facts and Rules.

2.2 Solver-augmented LMs

Solver-augmented LMs excel in deductive reasoning tasks. As shown in Figure 1(a), they can be generally divided into two stages: *Problem Formulation* and *Symbolic Reasoning*. In *Problem Formulation* stage, an LM is used to parse an NL logical question into symbolic representation (Figure 2). The process can be accomplished by providing LM with detailed instructions about the grammar of `Prolog`, alongside a few demonstrations as in-context examples. The LM is expected to identify the symbolic Facts, Rules, and Query from the NL question following the instructions; In *Symbolic Reasoning* stage, a solver takes in the symbolic representation obtained in the previous stage and conducts symbolic reasoning. The reasoning process of external off-the-shelf solver, e.g., pyke expert system (Frederiksen, 2008), is deterministic and invisible. Then, the truth value of the parsed Query, which is the only output of the solver, can be treated as the answer to the given question.

2.3 Analysis on the Parsing Successful Rate

Once the solver-augmented LMs correctly formulate the problem, the answers obtained through symbolic reasoning will be faithful, attributed to the deterministic nature of the solver. However, this heavily relies on the in-context learning (ICL) capabilities of LMs. Thus, we first calculate the parsing successful rate of three selected open-source LLMs on two deductive reasoning datasets in Table 1. Firstly, we observe that CodeLlama-13B-Base (`CodeLlama-13b-hf`) is unable to effectively conduct NL-to-SL parsing due to the limited ICL capabilities in NL. Then we can find that replacing the Base model with the Instruct version (`CodeLlama-13b-Instruct-hf`) can alleviate this issue, which may be attributed to the fact that the Instruct version is further fine-tuned with approx. 5B tokens to better follow human instructions. Overall, open-source LMs still exhibit parsing performance below expectations in certain situations.

Model	ProofWriter	PrOntoQA
Vicuna-13B	17.00	40.80
CodeLlama-13B-Base	0.33	0.40
CodeLlama-13B-Instruct	71.33	77.80

Table 1: Parsing successful rate (%) of our selected open-source LLMs on two deductive reasoning datasets.

3 LoGiPT

In this paper, we aim to remove the dependency on unsatisfactory NL-to-SL parsing and present a novel LM, LoGiPT instructed to imitate the logical reasoning process of Solvers for deductive reasoning tasks. To achieve this, we first reveal the solver reasoning process when solving logical problems (§3.1). Then, we construct a solver-derived instruction-tuning dataset, comprising NL logical questions and corresponding SL reasoning steps (§3.2). Finally, we fine-tune open-source LLMs using this dataset to develop LoGiPT (§3.3).

3.1 Revealing the Solver Reasoning Process

Before operating on the solvers, we first adopt `gpt-4` as the problem formulator for NL-to-SL parsing with instructions about the grammar and few-shot demonstrations[2], and obtain the SL representations of all training questions of the given logical datasets. Then, consistent with solver-augmented LMs, we adopt pyke as the symbolic solver in this work that can make inferences using the `Prolog` SL. Given a logical question, pyke first sets up a knowledge base and injects all known

[1]In this paper, the term 'Query' refers to a specific sentence of statement or comment, while 'question' is used in a broader sense to denote the description of a logical problem.

[2]Detailed instructions for parsing are in Appendix A,B.

```
Turn-1 (from human):
Task Description: Given a statement containing logical facts
and rules in natural languages as context, the task is to:
1. define all the predicates in logic forms from the context
2. parse the statement into known facts and rules based on the
defined predicates
3. begin reasoning to obtain all implied facts

Context:
Charlie is green. Charlie is white. Dave is blue. Dave is
furry. Dave is white. Fiona is blue. Fiona is green. Fiona is
nice. Fiona is rough. Fiona is white. All furry people are
quiet. All green, white people are nice. If someone is blue
and quiet then they are green.

Reasoning:
```

```
Turn-2 (from LM):
We first define following predicates and corresponding natural
language explanations:
Green($x, bool) ::: Is x green?
(... more predicates here ...)

We have following known facts from the context:
Green('Charlie', True)
(... more facts here ...)

We have following known rules from the context:
rule1: Furry($x, True) → Quiet($x, True)
rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
rule3: Blue($x, True) ∧ Quiet($x, True) → Green($x, True)

Now begin reasoning to obtain all implied facts:

Use rule1: Furry($x, True) → Quiet($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Quiet('Dave', True)
Unbind $x
Finish implied with rule1

Use rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
Bind $x to 'Charlie'
Obtain a new implied fact: Nice('Charlie', True)
Unbind $x
Bind $x to 'Fiona'
Obtain an already known or implied fact: Nice('Fiona', True)
Unbind $x
Finish implied with rule2

Use rule3: Blue($x, True) ∧ Quiet($x, True) → Green($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Green('Dave', True)
Reuse rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Nice('Dave', True)
Finish implied with rule2
Unbind $x
Bind $x to 'Fiona'
Unbind $x
Finish implied with rule3

Finally, we obtain following implied facts:
Green('Dave', True)
(... more facts here ...)

Finish reasoning
```

```
Turn-3 (from human):
Question: Based on the above information, is the following
comment true, false, or unknown? Dave is not green.

Options:
A) True
B) False
C) Unknown

Now you should:
1. reason first carefully with the above known facts and
implied facts
2. then output your answer strictly in the format of "So, the
correct option is: X". The X should be replaced by "A) True",
"B) False" or "C) Unknown"
```

```
Turn-4 (from LM):
The comment can be parsed into: Green('Dave', False)
According to the implied fact: Green('Dave', True)
The comment contradicts the fact.
So, the correct option is: B) False
```

Figure 3: A comprehensive 4-turn training example of our instruction-tuning data. We highlight the initial occurrences of each functionality described in §3.1 using the corresponding colors. We omit some predicates and Facts in Turn-2 to save space. *Note: The figure is best viewed in color.*

Facts and Rules (Figure 2) from solver's inputs. Then, it iteratively applies Rules on already known or implied Facts, aiming at obtaining more implied Facts until the Query is proved or disproved.

The reasoning process executed by pyke solver is invisible to users and solver-augmented LMs use the solver as a *black-box*. We hypothesis the "chain-of-thought" reasoning process of the solver is valuable and LLMs are able to learn from it. To this end, we first modify the source code of pyke[3] to achieve the following functionalities:

1. For each application of a Rule, explicitly state the Rule being `'Used'`, or `'Reused'` if the Rule has been applied before.
2. When finishing the application of a Rule, explicitly state the `'Finish'` action.
3. When assigning a value (e.g., an entity) to a variable (e.g., $x) within a Fact in a Rule, explicitly specify the variable being assigned using `'Bind'` and its corresponding value.
4. When the variable assignment is complete, provide an explicit indication via `'Unbind'`.
5. When obtaining a new implied Fact, explicitly state the `'New Fact obtained'`. If this Fact is an `'Already known or implied Fact'`, this should also be noted explicitly.
6. Upon the completion of reasoning, explicitly display `'All newly implied Facts'` in the knowledge base.

With the above instructions, we can obtain the revealed solver's reasoning process for the construction of training data. We also highlighted the initial occurrences of each functionality using the corresponding colors in Figure 3 (Turn-2), where a case will be described in detail in the next section.

3.2 Constructing the Instruction-tuning Data

However, as previously mentioned, we cannot guarantee that LMs can complete the NL-to-SL parsing on any arbitrary questions. To this end, we first filter out all unsuccessfully parsed training cases that cannot be executed by pyke. Then we reorganize and refine the filtered training data to enhance the interpretability of the solver-derived reason-

[3]https://pyke.sourceforge.net/

ing steps. For each case, we divide the reasoning process into four conversational turns (Turn-1&3 for human and Turn-2&4 for LM), which will be described elaborately in the following paragraphs. We also provide a comprehensive training example of our instruction-tuning data[4] in Figure 3, and the full version is also included in Appendix C.

Turn-1: Instructions & NL logical Context. For each NL logical question within the training set, we begin by stripping away the specific Query statement while retaining the question Context and subsequently integrating it with elaborately crafted instructions. Taking the case in Figure 3 as an example, we temporarily exclude the Query 'Dave is not green' from the 'Context' field. Here, we only consider Query-agnostic question description to ensure that LMs initially focus on the logical background itself. This is because sometimes the ground-truth answer is "Unknown" (e.g., cases in ProofWriter). The truth value of the Query cannot be inferred from the Context, and therefore we need to deduce all implied Facts first.

Turn-2: Query-agnostic Solver-derived Reasoning. As we have acquired the solver's symbolic reasoning data in the revealing phase, our goal in Turn-2 is to further refine and enhance the reasoning process to achieve a more readable form of the solver's reasoning process. Specifically, for each question, we first define all necessary predicates and append the corresponding NL explanations. Then we list the known Facts and Rules extracted from the Context with interleaved NL comments. After that, we represent the application of each Rule by utilizing separate blocks, line by line. We strive to preserve as many solver actions as possible, such as 'Binding' and 'Unbinding', as well as the acquisition of new implied Facts, and so forth. Noting that this information has already been obtained during the revealing phase, we focus on the refinement of the solver-derived reasoning process. Finally, we enumerate all newly implied Facts to enable the model to perform an interim review.

Turn-3: Query & Answering Instructions. In Turn-3, we present instructions for answering a given Query. Following prior works (Ceri et al., 1989; Tafjord et al., 2021), a Query can be considered true within a certain logical context if it is explicitly mentioned or if it can be implied through several Rule applications. To handle negation, we consider two distinct assumptions: 1) the open-world assumption (OWA) that treats any fact that cannot be provable as special truth value "unknown"; 2) the closed-world assumption (CWA) where any fact not provable is assumed "false". Following both assumptions, we adjust the answering instructions, particularly the "Options" part.

Turn-4: Query-based Reasoning & Formatted Answer. In final Turn-4, we compare the parsed Query with all known Facts and implied Facts, expecting model to perform basic language inference and generate answer options in desired format.

3.3 Fine-tuning Open-source LLMs

After obtaining the refined deductive reasoning instruction-tuning dataset, we can fine-tune open-source LLMs with the expectation that the trained model (i.e., LoGiPT) can possess reasoning abilities similar to those of solvers. Consequently, for any given Query, we can bypass the syntax or grammatical errors derived from NL-to-SL parsing by directly generating answers with LoGiPT.

4 Experiments

We construct our instruction-tuning data on the training sets of two public deductive reasoning datasets and evaluate LoGiPT on their test sets.

4.1 Datasets

ProofWriter (Tafjord et al., 2021) is a commonly employed dataset for deductive reasoning. Following Pan et al. (2023), we adopt the open-world assumption (OWA) subset where the answer of each example is one of {*True*, *False*, *Unknown*}. The original dataset is partitioned into 5 subsets where each part requiring 0, $\leq$1, $\leq$2, $\leq$3, and $\leq$5 hops of reasoning, respectively. For evaluation, we adopted the version provided by Pan et al. (2023), which comprises 600 samples from the most challenging 5-hop subsets with balanced label distribution. For training, we merged all training subsets and obtained 41,433 examples after construction.

PrOntoQA (Saparov and He, 2023) is a synthetic logical reasoning dataset created recently to test the general deductive reasoning capacity of LLMs. We adopt the hardest *fictional characters* version following Pan et al. (2023) where the entities of Facts are fictional concept names (e.g., 'wumpus' instead of 'cat'), to avoid any confounding effects

[4] In the original case, the Query is 'Charlie is not green.'. We replace it with 'Dave is not green.' for better illustration.

Model	Prompting Methods	ProofWriter	PrOntoQA
Random Answering	-	33.33	50.00
closed-source LMs			
ChatGPT (`gpt-3.5-turbo`)	Few-shot Standard	35.50	47.40
ChatGPT (`gpt-3.5-turbo`)	Few-shot CoT	49.17	67.80
GPT-3.5 (`text-davinci-003`)	Few-shot Standard	36.16	51.80
GPT-3.5 (`text-davinci-003`)	Few-shot CoT	48.33	83.00
GPT-4 (`gpt-4`)	Few-shot Standard	52.67	77.40
GPT-4 (`gpt-4`)	Few-shot CoT	68.11	<u>98.79</u>
open-source LMs			
Vicuna-13B (`vicuna-13b-v1.5-16k`)	Few-shot Standard	35.50	53.80
Vicuna-13B (`vicuna-13b-v1.5-16k`)	Few-shot CoT	41.50	37.40
CodeLlama-13B-Base (`CodeLlama-13b-hf`)	Few-shot Standard	0.00	0.00
CodeLlama-13B-Base (`CodeLlama-13b-hf`)	Few-shot CoT	36.00	50.00
CodeLlama-13B-Instruct (`CodeLlama-13b-Instruct-hf`)	Few-shot Standard	36.83	52.20
CodeLlama-13B-Instruct (`CodeLlama-13b-Instruct-hf`)	Few-shot CoT	32.67	66.40
solver-argumented LMs			
LogicLM (`gpt-3.5-turbo`)	Few-shot CoT	58.33	61.00
LogicLM (`text-davinci-003`)	Few-shot CoT	71.45	85.00
LogicLM (`gpt-4`)	Few-shot CoT	79.66	83.20
ours			
LoGiPT (`vicuna-13b-v1.5-16k`)	No ICD Provided	81.17	**96.40**
LoGiPT (`CodeLlama-13b-hf`)	No ICD Provided	<u>**89.50**</u>	95.60
LoGiPT (`CodeLlama-13b-Instruct-hf`)	No ICD Provided	81.67	96.20

Table 2: Main results on two evaluation datasets. The best results of LoGiPT are in **bold** and the best results within each dataset are <u>underlined</u>. 'ICD' denotes 'In-Context Demonstrations'.

from knowledge acquired during the pretraining phase. Similar to ProofWriter, PrOntoQA is organized into several subsets based on the number of required reasoning steps. We use the hardest 5-hop subset for evaluation, which comprises 500 samples. Contrary to ProofWriter, PrOntoQA is in a closed-world assumption (CWA) subset where the answer of each example is one of $\{$*True*, *False*$\}$. For training, we merely merge all subsets and obtain 15,940 training cases after filtering out syntax-invalid ones.

4.2 Baselines

Closed-source LMs: We include the ChatGPT (`gpt-3.5-turbo`) (OpenAI, 2022), GPT-3.5 (`text-davinci-003`) (Ouyang et al., 2022) and GPT-4 (`gpt-4`) (OpenAI, 2023) as closed-source LMs for evaluation following Pan et al. (2023).

Open-source LMs: We also evaluate open-source LMs for research community. Specifically, we choose Vicuna-13B (`vicuna-13b-v1.5-16k`), a chatbot trained by fine-tuning LLaMA-2 (Touvron et al., 2023b) on user-shared conversations collected from ShareGPT, and CodeLlama-13B, foundation models for code tasks. We select the base version (`CodeLlama-13b-hf`), and instruction fine-tuned version (`CodeLlama-13b-Instruct-hf`).

Solver-argumented LMs: Finally, we compare our model against the solver-argumented LMs. We focus on the representative LogicLM (Pan et al., 2023) with underlying LLMs ChatGPT (`gpt-3.5-turbo`), GPT-3.5 (`text-davinci-003`) and GPT-4 (`gpt-4`), which serve as the state-of-the-art deductive reasoning methods.

Prompting Methods: Apart from the LMs, we also analyze two types of prompting methods for baselines: i) *Standard prompting* that uses ICL with few-shot demonstrations to directly answer the given question; ii) *Chain-of-Thought (CoT)* that utilizes step-by-step problem-solving process to generate explanations where few-shot demonstrations are also provided, and then outputs the final answer. For a fair comparison, we use the same in-context examples for NL-to-SL parsing when evaluating all models on the same dataset, consistent with Pan et al. (2023). To enhance the clarification, we also provide a specific baseline *'Random Answering'* that randomly outputs answer options. During the testing of LoGiPT, we did not employ few-shot demonstrations for in-context learning.

4.3 Implementation Details

In fine-tuning, we use a batch size of 32 per GPU and a learning rate of 1e-5 for all open-source LMs. We train our model on 8 Nvidia A100-80G GPUs with DeepSpeed ZeRO-3 (Rasley et al., 2020) for 12 hours on 2 epochs. For reproducibility, we use greedy decoding and set the temperature to 0 and the maximum context length to 8192. As for baselines, we strictly follow the setting of Pan et al. (2023). Given that all instances are presented in the form of multiple-choice questions, we assess the model performance by the accuracy of selecting the correct answer option.

4.4 Main Results

We report the results of LoGiPT and baselines on Table 2 and have following main findings:

1) When prompting with few-shot examples, open-source LMs exhibit notably poor deductive reasoning capabilities, with their outputs closed to random answering. Even the Standard prompting models of ChatGPT (`gpt-3.5-turbo`) and GPT-3.5 (`text-davinci-003`) exhibit a similar performance to random answering. This once again demonstrates that it is considerably difficult for many LLMs to solve logical reasoning tasks.

2) LoGiPT is significantly superior to the state-of-the-art solver-augmented LMs by a large margin on both deductive reasoning benchmarks. In ProofWriter, our best-performing model, LoGiPT (`CodeLlama-13b-hf`), outperforms the currently state-of-the-art LogicLM (`gpt-4`) by an absolute improvement of 9.84%. Meanwhile, in PrOntoQA, our best-performing model LoGiPT (`vicuna-13b-v1.5-16k`) exhibits an even higher absolute improvement of 13.20% than LogicLM (`gpt-4`). The improvement of both LoGiPT models is statistically significant compared with the baseline LogicLM (`gpt-4`) (t-test, p-value < 0.05). This indicates that our approach is better than the pipeline of problem formulation first and then reasoning with solvers, and fine-tuning with solver-derived reasoning data can facilitate the deductive reasoning capacity of LMs.

3) LoGiPT significantly outperforms all selected open/closed-source LMs on both datasets, except for the CoT experiment on the PrOntoQA data where LoGiPT achieves comparable results with GPT-4 CoT. This is surprising considering that our underlying open-source LMs are merely 13B parameters in size. As for the baseline experiments of GPT-4, our performance on ProofWriter also significantly surpasses that of GPT-4's Standard and CoT prompting versions, as well as the Standard version of PrOntoQA. These results further demonstrate that open-source LMs, when coupled with solver-simulated reasoning capacity, can achieve performance on par with or even superior to closed-source GPT models.

4) The accuracy of CodeLlama-13B-Base with Standard prompting was 0.00, and the performance of the CoT version was close to random answering. By examining the outputs, we found that this is due to the CodeLlama-13B-Base's inability to follow the provided few-shot demonstrations, resulting in outputting no answering options. The introduction of the Instruct version of CodeLlama-13B mitigates this issue to some extent. However, after training with LoGiPT, the CodeLlama models far less encounter this issue (i.e., following the right answering format in both test sets) and even achieve better performance than the Vicuna version of LoGiPT. This demonstrates the potential of code foundation models in logical reasoning tasks, consistent with the finding on prior work (Yue et al., 2023).

Model	Accuracy
LoGiPT (`vicuna-13b-v1.5-16k`)	81.17
+ (w/o ‘`unbind`’ statements)	80.67
+ (w/o ‘`fail & backtrack`’ statements)	84.00
+ (w/ NL representation)	66.33
LoGiPT (`CodeLlama-13b-hf`)	89.50
+ (w/o ‘`unbind`’ statements)	93.33
+ (w/o ‘`fail & backtrack`’ statements)	87.17
+ (w/ NL representation)	52.33
LoGiPT (`CodeLlama-13b-Instruct-hf`)	81.67
+ (w/o ‘`unbind`’ statements)	79.00
+ (w/o ‘`fail & backtrack`’ statements)	84.83
+ (w/ NL representation)	66.33

Table 3: The accuracy of the variations on solver-derived reasoning format, and replacing SL representations with NL on ProofWriter. The best results on each LMs are underlined.

4.5 Further Analysis

Impact of Solver-derived Reasoning Formats We further investigate the impact of different solver-derived reasoning formats on the model's performance. Specifically, we consider the following format variations: 1) *w/o ‘unbind’ statements* that we remove all `‘Unbind’` statements from **Turn-2** to investigate the utility of the explicit retention of this action from the solver; 2) *w/o 'fail & backtrack' statements* that we removing all `'Fail & backtrack'` statements from **Turn-2**. During the solver's reasoning process, it is expected to encounter situations in which, after binding a value, the solver realizes that not all premises are satisfied (e.g., `'Fiona is blue'` but `'Fiona is not quiet'` for application of Rule3 in Figure 3). Consequently, a `'Fail & backtrack'` operation occurs (highlighted in color in Figure 3). We explore the effectiveness of explicitly stating these operations.

Table 3 presents the accuracy on ProofWriter where several observations can be made: 1) regardless of using the default format, removing `'Unbind'` statements, or removing `'Fail & backtrack'` statements, it can not be determined which format guarantees the optimal results. To retain the maximum amount of action information that the solver can provide, we still adopt the default settings in LoGIPT; 2) whether `'Unbind'` statements are removed or `'Fail & backtrack'` statements are removed, there is always an experiment under each open-source LMs that can surpass the default LoGIPT results. This further enhances the best performance of LoGIPT shown in Table 2.

Impact of SL Reasoning Representations We are also curious about the impact of SL reasoning representations. Thus, we include additional experiments in Table 3, denoted as *w/ NL representation* that we re-translate the symbolic representation (e.g., `Green('Charlie', True)`) back to its original NL version (e.g., `Charlie is green.`) and replace the original symbolic representation in **Turn-2**. From the table, we can find that replacing SL representations with NL results in a significant decrease in model performance, further emphasizing that symbolic representations are superior to NL representations in deductive reasoning tasks.

Case Study We also present three typical instances of symbolic reasoning steps generated by LoGIPT (`CodeLlama-13b-hf`) from ProofWriter in Figure 5,6,7 of Appendix E. Overall, after fine-tuning with our constructed solver-derived instruction-tuning dataset, LoGIPT demonstrates a satisfactory ability to mimic the reasoning process of logical solvers, as illustrated in Figure 5. Nevertheless, there are still certain difficulties in rigorously conducting logical reasoning entirely in accordance with the rationality of the solvers.

In Figure 6, we observe that the symbolic reasoning process in Turn-2 is accurate. However, in Turn-4, the model erroneously identifies a Fact that aligned with the predicate of the Query ('Young'), yet it notes inconsistency in their first arguments ('Bear' and 'Cat'). Then, the model erroneously considers them contradictory, leading to the incorrect answer option 'B) False'. While in this case, the semantic contradiction occurs only when there is a conflict in boolean values (i.e., Young('Bear', False)). So the correct option should be 'C) Unknown'. Another typical error case is shown in Figure 7, where the model exhibits an erroneous backtracking behavior in Turn-2 (specifically during the reuse of rules) and omits the necessary steps for reasoning utilizing rule7. Consequently, in Turn-4 the essential Fact, Chases('Bear', 'BaldEagle', True), is missing, leading to an incorrect conclusion.

We also explore merging data from different reasoning assumptions in Appendix D.

5 Related Work

Logical Reasoning with LMs. Recent efforts in adapting LLMs for logical reasoning tasks generally adopt direct fine-tuning specialized modules (Clark et al., 2020; Tafjord et al., 2021, 2022; Yang et al., 2022) or ICL (Zhou et al., 2022; Lyu et al., 2023; Ling et al., 2023; Liu et al., 2023c), where reasoning in NL is used by both groups of methods. Fine-tuning approaches involve training the full model or specialized modules, enhancing LLMs with module-level logical reasoning skills like proof, enumeration, and abduction (Tafjord et al., 2021). The ICL approaches create specific prompts to encourage LLMs' step-by-step reasoning skills. Common methods encompass CoT prompting (Wei et al., 2022b; Chen et al., 2023), which produces explanations before delivering a final answer, and least-to-most prompting (Zhou et al., 2022), which deconstructs a problem into simpler components that can be resolved individually. Some recent work has focused on combining neural networks with symbolic reasoning (Tian et al., 2022; Pryor et al., 2022; Pan et al., 2023; Yang et al., 2023), especially the solver-augmented LMs that parse NL questions into symbolic representations, then utilizing external logical solvers for answering. Despite their impressive performance, parsing errors can lead to solver execution failure and logical question-answering issues.

Augmented LMs for Reasoning. Recent work has begun to augment LMs to overcome their inherent limitations such as the incapacity to access up-to-date information or conduct accurate mathematical reasoning. They augment with external tools and resources, such as the information retriever (Shi et al., 2023; Lazaridou et al., 2022), planner (Liu et al., 2023a) and other pre-trained models (Shen et al., 2023). Specifically, to enhance the reasoning capacity, recent work resort to external off-the-shelf **Solvers** including programmatic interpreters (Chen et al., 2022; Gao et al., 2023), satisfiability solvers (Ye et al., 2023), logical solvers (Pan et al., 2023) or their hybrids (Poesia et al., 2023). Most of them utilize the LMs to parse the NL question to symbolic representations and then invoke solvers to reason in SL. In this paper, we concentrate on logical solvers, automated tools for validating the truth value of logical formulas.

6 Conclusion

In this paper, we propose a novel LoGiPT that can directly act as a logical solver for deductive reasoning tasks. LoGiPT can output all facts implied from NL logical questions, while bypassing the syntax or grammatical errors derived from NL-to-SL parsing of solver-augmented LMs. We conducted numerous analytical experiments on two public deductive reasoning benchmarks. Evaluation results show that LoGiPT can significantly outperform state-of-the-art solver-augmented LMs, and surpass or be comparable with few-shot prompting methods on competitive LLMs like GPT-4.

Limitations

Besides its merits, this work still has two main limitations that could be further explored. Firstly, in this paper, we mainly focus on exploring how to empower LMs to directly learn or emulate the reasoning process of logical solvers. Hence, we choose deductive reasoning as a starting point for research. Our proposed LoGiPT is also mainly used for deductive reasoning and thus cannot handle more complex logical problems such as constraint satisfaction problems and first-order logic reasoning problems. Plus, we employ forward-chaining reasoning while backward-chaining is also an efficient reasoning approach (Kazemi et al., 2023). Fortunately, our approach is not confined solely to specific logical tasks. Therefore, in the future, we aim to endow the LMs with more reasoning abilities by revealing the reasoning process of corresponding solvers for other logical tasks.

Secondly, we observe limited diversity in the Rule formats of the two datasets utilized in this paper (e.g., in PrOntoQA, the majority of Rules are in the format of 'Every/Each `A` is (not) `B`' or '`A` are (not) `B`'. Please refer to Appendix D for more details). Consequently, LMs trained on our constructed instruction-tuning data may not be able to generalize to more complex logical Rules. As a potential solution, future works could explore the enhancement of the style/genre of logical context by paraphrasing with prompting small-scale LMs, or augmenting training data with synthetic data.

Ethics Statement

This paper introduces LoGiPT, a novel LM designed to mimic the reasoning process of logical solvers, enabling it to solve deductive reasoning tasks. The data-collecting APIs and closed- or open-source LMs are only strictly used for academic purposes. The proposed method does not introduce ethical or social bias into the collected data.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by the National Key R&D Program of China (No.2022YFC3301900).

References

Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. 1989. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166.

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

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models. *arXiv preprint arXiv:2305.14323*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages

3882–3890. International Joint Conferences on Artificial Intelligence Organization. Main track.

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

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

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Google. 2023. Google bard. *https://bard.google.com/*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. LAMBADA: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6547–6568, Toronto, Canada. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*.

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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2304.03439*.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023c. LogiCoT: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921, Singapore. Association for Computational Linguistics.

Man Luo, Shrinidhi Kumbhar, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, Chitta Baral, et al. 2023. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models. *arXiv preprint arXiv:2310.00836*.

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

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *https://openai.com/blog/chatgpt/*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Gabriel Poesia, Kanishk Gandhi, Eric Zelikman, and Noah D Goodman. 2023. Certified reasoning with language models. *arXiv preprint arXiv:2306.04031*.

David L 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. *arXiv preprint arXiv:2205.14268*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th*

ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3505–3506.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

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

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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2301.12652*.

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.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2078–2093, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2022. Weakly supervised neural symbolic learning for cognitive tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5888–5896.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 89–105, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sen Yang, Xin Li, Leyang Cui, Lidong Bing, and Wai Lam. 2023. Neuro-symbolic integration brings causal and reliable reasoning proofs. *arXiv preprint arXiv:2311.09802*.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satisfiability-aided language models using declarative prompting. *arXiv preprint arXiv:2305.09656*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

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

A Instructions for NL-to-SL Parsing on ProofWriter

Task Description: You are given a problem description and a question. The task is to:
1) define all the predicates in the problem
2) parse the problem into logic rules based on the defined predicates
3) write all the facts mentioned in the problem
4) parse the question into the logic form

Problem:
Anne is quiet. Erin is furry. *(... 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 ...)
Young($x, bool) ::: Is x young?

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

Rules:
Young($x, True) >>> Furry($x, True) ::: Young people are furry.
Quite(Anne, True) >>> Red($x, True) ::: If Anne is quiet then Anne is red.
(... more rules here ...)
Red($x, True) >>> Young($x, True) ::: All red people are young.

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

Problem:
(new problem here)
Question:
(new question here)

B Instructions for NL-to-SL Parsing on PrOntoQA

Task Description: You are given a problem description and a question. The task is to:
1) define all the predicates in the problem
2) parse the problem into logic rules based on the defined predicates
3) write all the facts mentioned in the problem
4) parse the question into the logic form

Problem:
Each jompus is fruity. Every jompus is a wumpus. *(... more context here ...)* Alex is a tumpus.

Question:
True or false: Alex is not shy.

Predicates:
Jompus($x, bool) ::: Does x belong to Jompuses?
Fruity($x, bool) ::: Is x fruity?
(... more predicates here ...)
Liquid($x, bool) ::: Is x liquid?
Zumpus($x, bool) ::: Does x belong to Zumpuses?

Facts:
Tumpus(Alex, True) ::: Alex is a tumpus.

Rules:
Jompus($x, True) >>> Fruity($x, True) ::: Each jompus is fruity.
Jompus($x, True) >>> Wumpus($x, True) ::: Every jompus is a wumpus.
(... more rules here ...)
Rompus($x, True) >>> Liquid($x, True) ::: Each rompus is liquid.
Rompus($x, True) >>> Zumpus($x, True) ::: Rompuses are zumpuses.

Query:
Shy(Alex, False) ::: Alex is not shy.

Problem:
(new problem here)
Question:
(new question here)

C A Comprehensive Training Example of our Instruction-tuning Data

We present in Figure 4 the full version of the comprehensive 4-turn training example of our instruction-tuning data shown in Figure 3.

D Effectiveness of Merging Data from Different Reasoning Assumptions

Since ProofWriter is an open-world assumption and PrOntoQA is labeled within a closed-world assumption, we also perform a further investigation on whether both reasoning assumptions can benefit each other. Specifically, we first merge both constructed training data and then test LoGiPT on each test set. The experimental results are shown

```
Turn-1 (from human):
Task Description: Given a statement containing logical facts
and rules in natural languages as context, the task is to:
1. define all the predicates in logic forms from the context
2. parse the statement into known facts and rules based on the
defined predicates
3. begin reasoning to obtain all implied facts

Context:
Charlie is green. Charlie is white. Dave is blue. Dave is
furry. Dave is white. Fiona is blue. Fiona is green. Fiona is
nice. Fiona is rough. Fiona is white. All furry people are
quiet. All green, white people are nice. If someone is blue
and quiet then they are green.

Reasoning:
```

```
Turn-2 (from LM):
We first define following predicates and corresponding natural
language explanations:
Green($x, bool) ::: Is x green?
White($x, bool) ::: Is x white?
Blue($x, bool) ::: Is x blue?
Furry($x, bool) ::: Is x furry?
Nice($x, bool) ::: Is x nice?
Rough($x, bool) ::: Is x rough?
Quiet($x, bool) ::: Is x quiet?

We have following known facts from the context:
Green('Charlie', True)
Green('Fiona', True)
White('Charlie', True)
White('Dave', True)
White('Fiona', True)
Blue('Dave', True)
Blue('Fiona', True)
Furry('Dave', True)
Nice('Fiona', True)
Rough('Fiona', True)

We have following known rules from the context:
rule1: Furry($x, True) → Quiet($x, True)
rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
rule3: Blue($x, True) ∧ Quiet($x, True) → Green($x, True)

Now begin reasoning to obtain all implied facts:

Use rule1: Furry($x, True) → Quiet($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Quiet('Dave', True)
Unbind $x
Finish implied with rule1

Use rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
Bind $x to 'Charlie'
Obtain a new implied fact: Nice('Charlie', True)
Unbind $x
Bind $x to 'Fiona'
Obtain an already known or implied fact: Nice('Fiona', True)
Unbind $x
Finish implied with rule2

Use rule3: Blue($x, True) ∧ Quiet($x, True) → Green($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Green('Dave', True)
Reuse rule2: Green($x, True) ∧ White($x, True) → Nice($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Nice('Dave', True)
Finish implied with rule2
Unbind $x
Bind $x to 'Fiona'
Unbind $x
Finish implied with rule3

Finally, we obtain following implied facts:
Green('Dave', True)
Nice('Charlie', True)
Nice('Dave', True)
Quiet('Dave', True)

Finish reasoning
```

```
Turn-3 (from human):
Question: Based on the above information, is the following
comment true, false, or unknown? Dave is not green.

Options:
A) True
B) False
C) Unknown

Now you should:
1. reason first carefully with the above known facts and
implied facts
2. then output your answer strictly in the format of "So, the
correct option is: X". The X should be replaced by "A) True",
"B) False" or "C) Unknown"
```

```
Turn-4 (from LM):
The comment can be parsed into: Green('Dave', False)
According to the implied fact: Green('Dave', True)
The comment contradicts the fact.
So, the correct option is: B) False
```

Figure 4: The full version of the comprehensive 4-turn training example of our instruction-tuning data shown in Figure 3.

Train set	Test Set	VCN	CLB	CLI
PrOntoQA	PrOntoQA	96.40	95.60	96.20
Both	PrOntoQA	91.00	87.00	89.00
Both (Reformat)	PrOntoQA	90.00	87.00	77.80
ProofWriter	ProofWriter	81.17	89.50	81.67
Both	ProofWriter	79.33	87.17	79.67
Both (Reformat)	ProofWriter	79.00	90.83	84.50

Table 4: The accuracy of LoGiPT trained with merged data and tested on single data with different underlying LMs. 'VCN', 'CLB', and 'CLI' respectively represent Vicuna-13B, CodeLlama-13B-Base, and CodeLlama-13B-Instruct. 'Both' means 'ProofWriter + PrOntoQA'.

in Table 4. We can conclude that if we directly mix the two types of data for training, the results on their respective test sets will be slightly lower than those obtained from training solely on their respective datasets. Therefore, we conducted an in-depth analysis of the underlying reasons and observed that in PrOntoQA, the majority of Rules are in the format of 'Every/Each A is (not) B' or 'A are (not) B'. While in ProofWriter, the predominant structure of Rules consists of: 'If someone is A, then they are B' or 'If something is A, then it is B'. Therefore, we conducted an additional set of experiments in which the Rule format of two training sets was randomly reformatted into the four aforementioned types using regular expression (denoted as 'Both (Reformat)'). Then, we test the model on the original test sets. We can observe that by employing this approach, the code models yield improved performance on ProofWriter. Thus, the style/genre of logical context must also be taken into consideration to maximize the efficacy of transfer learning in logical reasoning.

E Typical Instances of Generated Symbolic Reasoning Steps

We also showcase three typical instances of symbolic reasoning steps generated by LoGiPT (`CodeLlama-13b-hf`) from the test set of ProofWriter in Figure 5, 6 and 7.

```
Turn-1 (from human):
Task Description: Given a statement containing logical facts
and rules in natural languages as context, the task is to:
1. define all the predicates in logic forms from the context
2. parse the statement into known facts and rules based on the
defined predicates
3. begin reasoning to obtain all implied facts

Context:
Bob is big. Bob is green. Dave is kind. Dave is white. Erin is
green. Erin is kind. Erin is smart. Fiona is big. Fiona is
blue. Fiona is furry. White, blue things are furry. All big,
white things are smart. Furry things are green. Kind things
are blue. All green things are furry. Green things are big.

Reasoning:
```

```
Turn-2 (from LM):
We first define following predicates and corresponding natural
language explanations:
Big($x, bool) ::: Is x big?
Green($x, bool) ::: Is x green?
Kind($x, bool) ::: Is x kind?
White($x, bool) ::: Is x white?
Smart($x, bool) ::: Is x smart?
Blue($x, bool) ::: Is x blue?
Furry($x, bool) ::: Is x furry?

We have following known facts from the context:
Big('Bob', True)
Big('Fiona', True)
Green('Bob', True)
Green('Erin', True)
Kind('Dave', True)
Kind('Erin', True)
White('Dave', True)
Smart('Erin', True)
Blue('Fiona', True)
Furry('Fiona', True)

We have following known rules from the context:
rule1: White($x, True) ∧ Blue($x, True) → Furry($x, True)
rule2: Big($x, True) ∧ White($x, True) → Smart($x, True)
rule3: Furry($x, True) → Green($x, True)
rule4: Kind($x, True) → Blue($x, True)
rule5: Green($x, True) → Furry($x, True)
rule6: Green($x, True) → Big($x, True)

Now begin reasoning to obtain all implied facts:

Use rule3: Furry($x, True) → Green($x, True)
Bind $x to 'Fiona'
Obtain a new implied fact: Green('Fiona', True)
Unbind $x
Finish implied with rule3

Use rule4: Kind($x, True) → Blue($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Blue('Dave', True)
Reuse rule1: White($x, True) ∧ Blue($x, True) → Furry($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Furry('Dave', True)
Reuse rule3: Furry($x, True) → Green($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Green('Dave', True)
Finish implied with rule3
Finish implied with rule1
Unbind $x
Bind $x to 'Erin'
Obtain a new implied fact: Blue('Erin', True)
Reuse rule1: White($x, True) ∧ Blue($x, True) → Furry($x, True)
Bind $x to 'Erin'
Finish implied with rule1
Unbind $x
Finish implied with rule4

Use rule5: Green($x, True) → Furry($x, True)
Bind $x to 'Bob'
Obtain a new implied fact: Furry('Bob', True)
Reuse rule3: Furry($x, True) → Green($x, True)
Bind $x to 'Bob'
Obtain an already known or implied fact: Green('Bob', True)
Finish implied with rule3
Unbind $x
Bind $x to 'Erin'
Obtain a new implied fact: Furry('Erin', True)
Reuse rule3: Furry($x, True) → Green($x, True)
Bind $x to 'Erin'
Obtain an already known or implied fact: Green('Erin', True)
Finish implied with rule3
Unbind $x
Bind $x to 'Fiona'
Obtain an already known or implied fact: Furry('Fiona', True)
Unbind $x
Bind $x to 'Dave'
Obtain an already known or implied fact: Furry('Dave', True)
Unbind $x
Finish implied with rule5

Use rule6: Green($x, True) → Big($x, True)
Bind $x to 'Bob'
Obtain an already known or implied fact: Big('Bob', True)
Unbind $x
Bind $x to 'Erin'
Obtain a new implied fact: Big('Erin', True)
Reuse rule2: Big($x, True) ∧ White($x, True) → Smart($x, True)
Bind $x to 'Erin'
Finish implied with rule2
Unbind $x
Bind $x to 'Fiona'
Obtain an already known or implied fact: Big('Fiona', True)
Unbind $x
Bind $x to 'Dave'
Obtain a new implied fact: Big('Dave', True)
Reuse rule2: Big($x, True) ∧ White($x, True) → Smart($x, True)
Bind $x to 'Dave'
Obtain a new implied fact: Smart('Dave', True)
Finish implied with rule2
Unbind $x
Finish implied with rule6

Finally, we obtain following implied facts:
Big('Erin', True)
Big('Dave', True)
Green('Fiona', True)
Green('Dave', True)
Smart('Dave', True)
Blue('Dave', True)
Blue('Erin', True)
Furry('Dave', True)
Furry('Bob', True)
Furry('Erin', True)

Finish reasoning
```

```
Turn-3 (from human):
Question: Based on the above information, is the following
comment true, false, or unknown? Dave is not furry.

Options:
A) True
B) False
C) Unknown

Now you should:
1. reason first carefully with the above known facts and
implied facts
2. then output your answer strictly in the format of "So, the
correct option is: {X}". The {X} should be replaced by "A)
True", "B) False" or "C) Unknown"
```

```
Turn-4 (from LM):
The comment can be parsed into: Furry('Dave', False)
According to the implied fact: Furry('Dave', True)
The comment contradicts the fact.
So, the correct option is: B) False
```

Figure 5: An instance where LOGIPT (CodeLlama-13b-hf) successfully emulates a logical solver to conduct reasoning and arrive at the correct answer, originates from the test set of ProofWriter.

Turn-1 (from human):

```
Task Description: Given a statement containing logical facts
and rules in natural languages as context, the task is to:
1. define all the predicates in logic forms from the context
2. parse the statement into known facts and rules based on the
defined predicates
3. begin reasoning to obtain all implied facts

Context:
The bear is green. The bear likes the cat. The bear likes the
dog. The bear visits the dog. The cat is young. The cat sees
the bear. The cat sees the dog. The cat visits the bear. The
cat visits the mouse. The dog is round. The dog likes the
bear. The dog likes the mouse. The dog visits the mouse. The
mouse is big. The mouse is cold. The mouse is round. If
something visits the mouse and the mouse visits the dog then
it is cold. If something likes the cat then it visits the dog.
If something is cold then it likes the cat. If something is
green then it sees the dog. If something likes the mouse then
it sees the cat. If something is green and cold then it likes
the cat. If something is big and it visits the bear then the
bear is green.

Reasoning:
```

Turn-2 (from LM):

```
We first define following predicates and corresponding natural
language explanations:
Green($x, bool) ::: Is x green?
Likes($x, $y, bool) ::: Does x like y?
Visits($x, $y, bool) ::: Does x visit y?
Young($x, bool) ::: Is x young?
Sees($x, $y, bool) ::: Does x see y?
Round($x, bool) ::: Is x round?
Big($x, bool) ::: Is x big?
Cold($x, bool) ::: Is x cold?

We have following known facts from the context:
Green('Bear', True)
Likes('Bear', 'Cat', True)
Likes('Bear', 'Dog', True)
Likes('Dog', 'Bear', True)
Likes('Dog', 'Mouse', True)
Visits('Bear', 'Dog', True)
Visits('Cat', 'Bear', True)
Visits('Cat', 'Mouse', True)
Visits('Dog', 'Mouse', True)
Young('Cat', True)
Sees('Cat', 'Bear', True)
Sees('Cat', 'Dog', True)
Round('Dog', True)
Round('Mouse', True)
Big('Mouse', True)
Cold('Mouse', True)

We have following known rules from the context:
rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True) →
Cold($x, True)
rule2: Likes($x, Cat, True) → Visits($x, Dog, True)
rule3: Cold($x, True) → Likes($x, Cat, True)
rule4: Green($x, True) → Sees($x, Dog, True)
rule5: Likes($x, Mouse, True) → Sees($x, Cat, True)
rule6: Green($x, True) ∧ Cold($x, True) → Likes($x, Cat, True)
rule7: Big($x, True) ∧ Visits($x, Bear, True) → Green(Bear,
True)

Now begin reasoning to obtain all implied facts:

Use rule2: Likes($x, Cat, True) → Visits($x, Dog, True)
Bind $x to 'Bear'
Obtain an already known or implied fact: Visits('Bear', 'Dog',
True)
Unbind $x
Finish implied with rule2

Use rule3: Cold($x, True) → Likes($x, Cat, True)
Bind $x to 'Mouse'
Obtain a new implied fact: Likes('Mouse', 'Cat', True)
Reuse rule2: Likes($x, Cat, True) → Visits($x, Dog, True)
Bind $x to 'Mouse'
Obtain a new implied fact: Visits('Mouse', 'Dog', True)
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Bind $x to 'Mouse'
Finish implied with rule1
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Bind $x to 'Cat'
Obtain a new implied fact: Cold('Cat', True)
Reuse rule3: Cold($x, True) → Likes($x, Cat, True)
Bind $x to 'Cat'
Obtain a new implied fact: Likes('Cat', 'Cat', True)
Reuse rule2: Likes($x, Cat, True) → Visits($x, Dog, True)
Bind $x to 'Cat'
Obtain a new implied fact: Visits('Cat', 'Dog', True)
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Bind $x to 'Cat'
Finish implied with rule1
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Finish implied with rule1
Finish implied with rule2
Finish implied with rule3
Unbind $x
Bind $x to 'Dog'
Obtain a new implied fact: Cold('Dog', True)
Reuse rule3: Cold($x, True) → Likes($x, Cat, True)
Bind $x to 'Dog'
Obtain a new implied fact: Likes('Dog', 'Cat', True)
Reuse rule2: Likes($x, Cat, True) → Visits($x, Dog, True)
Bind $x to 'Dog'
Obtain a new implied fact: Visits('Dog', 'Dog', True)
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Bind $x to 'Dog'
Finish implied with rule1
Reuse rule1: Visits($x, Mouse, True) ∧ Visits(Mouse, Dog, True)
→ Cold($x, True)
Finish implied with rule1
Finish implied with rule2
Finish implied with rule3
Unbind $x
Finish implied with rule1
Unbind $x
Bind $x to 'Cat'
Obtain an already known or implied fact: Likes('Cat', 'Cat',
True)
Unbind $x
Bind $x to 'Dog'
Obtain an already known or implied fact: Likes('Dog', 'Cat',
True)
Unbind $x
Finish implied with rule3

Use rule4: Green($x, True) → Sees($x, Dog, True)
Bind $x to 'Bear'
Obtain a new implied fact: Sees('Bear', 'Dog', True)
Unbind $x
Finish implied with rule4

Use rule5: Likes($x, Mouse, True) → Sees($x, Cat, True)
Bind $x to 'Dog'
Obtain a new implied fact: Sees('Dog', 'Cat', True)
Unbind $x
Finish implied with rule5

Finally, we obtain following implied facts:
Likes('Mouse', 'Cat', True)
Likes('Cat', 'Cat', True)
Likes('Dog', 'Cat', True)
Visits('Mouse', 'Dog', True)
Visits('Cat', 'Dog', True)
Visits('Dog', 'Dog', True)
Sees('Bear', 'Dog', True)
Sees('Dog', 'Cat', True)
Cold('Cat', True)
Cold('Dog', True)

Finish reasoning
```

Turn-3 (from human):

```
Question: Based on the above information, is the following
comment true, false, or unknown? The bear is young.

Options:
A) True
B) False
C) Unknown

Now you should:
1. reason first carefully with the above known facts and
implied facts
2. then output your answer strictly in the format of "So, the
correct option is: {X}". The {X} should be replaced by "A)
True", "B) False" or "C) Unknown"
```

Turn-4 (from LM):

The comment can be parsed into: Young('Bear', True)
~~According to the known fact: Young('Cat', True)~~
~~The comment contradicts the fact.~~
~~So, the correct option is: B) False~~
According to the above known facts and implied facts, we can not determine whether the comment is true or false.
So, the correct option is: C) Unknown

Figure 6: An error case from the test set of ProofWriter where LoGiPT (CodeLlama-13b-hf) fails to identify the reference Fact to judge the truth value of the Query ('The bear is young'). The incorrect reasoning steps are marked in yellow, and the correct revisions or missing steps are marked in green.

Turn-1 (from human):
Task Description: Given a statement containing logical facts and rules in natural languages as context, the task is to:
1. define all the predicates in logic forms from the context
2. parse the statement into known facts and rules based on the defined predicates
3. begin reasoning to obtain all implied facts

Context:
The bald eagle is big. The bald eagle is not green. The bald eagle likes the bear. The bald eagle likes the squirrel. The bear chases the lion. The bear is green. The bear visits the bald eagle. The bear visits the squirrel. The lion is not green. The lion likes the bear. The lion visits the squirrel. The squirrel chases the bald eagle. The squirrel is cold. The squirrel visits the bald eagle. If someone is cold then they are not young. All big people are not green. If the lion is big and the lion is green then the lion chases the bald eagle. If someone chases the bald eagle then they like the bear. If someone is round and they visit the bald eagle then the bald eagle likes the squirrel. If someone visits the lion then they like the bald eagle. If the lion visits the squirrel and the squirrel chases the bear then the bear chases the bald eagle. If someone visits the lion then they are cold. If someone likes the bear then they chase the bear.

Reasoning:

Turn-2 (from LM):
We first define following predicates and corresponding natural language explanations:
Big($x, bool) ::: Is x big?
Green($x, bool) ::: Is x green?
Likes($x, $y, bool) ::: Does x like y?
Chases($x, $y, bool) ::: Does x chase y?
Visits($x, $y, bool) ::: Does x visit y?
Cold($x, bool) ::: Is x cold?
Young($x, bool) ::: Is x young?
Round($x, bool) ::: Is x round?

We have following known facts from the context:
Big('BaldEagle', True)
Green('BaldEagle', False)
Green('Bear', True)
Green('Lion', False)
Likes('BaldEagle', 'Bear', True)
Likes('BaldEagle', 'Squirrel', True)
Likes('Lion', 'Bear', True)
Chases('Bear', 'Lion', True)
Chases('Squirrel', 'BaldEagle', True)
Visits('Bear', 'BaldEagle', True)
Visits('Bear', 'Squirrel', True)
Visits('Lion', 'Squirrel', True)
Visits('Squirrel', 'BaldEagle', True)
Cold('Squirrel', True)

We have following known rules from the context:
rule1: Cold($x, True) → Young($x, False)
rule2: Big($x, True) → Green($x, False)
rule3: Big(Lion, True) ∧ Green(Lion, True) → Chases(Lion, BaldEagle, True)
rule4: Chases($x, BaldEagle, True) → Likes($x, Bear, True)
rule5: Round($x, True) ∧ Visits($x, BaldEagle, True) → Likes(BaldEagle, Squirrel, True)
rule6: Visits($x, Lion, True) → Likes($x, BaldEagle, True)
rule7: Visits(Lion, Squirrel, True) ∧ Chases(Squirrel, Bear, True) → Chases(Bear, BaldEagle, True)
rule8: Visits($x, Lion, True) → Cold($x, True)
rule9: Likes($x, Bear, True) → Chases($x, Bear, True)

Now begin reasoning to obtain all implied facts:

Use rule1: Cold($x, True) → Young($x, False)
Bind $x to 'Squirrel'
Obtain a new implied fact: Young('Squirrel', False)
Unbind $x
Finish implied with rule1

Use rule2: Big($x, True) → Green($x, False)
Bind $x to 'BaldEagle'
Obtain an already known or implied fact: Green('BaldEagle', False)
Unbind $x
Finish implied with rule2

Use rule4: Chases($x, BaldEagle, True) → Likes($x, Bear, True)
Bind $x to 'Squirrel'
Obtain a new implied fact: Likes('Squirrel', 'Bear', True)
Unbind $x
Finish implied with rule4

Use rule9: Likes($x, Bear, True) → Chases($x, Bear, True)
Bind $x to 'BaldEagle'
Obtain a new implied fact: Chases('BaldEagle', 'Bear', True)
~~Reuse rule4: Chases($x, BaldEagle, True) → Likes($x, Bear, True)~~
~~Bind $x to 'BaldEagle'~~
~~Finish implied with rule4~~
~~Reuse rule7: Visits(Lion, Squirrel, True) ∧ Chases(Squirrel, Bear, True) → Chases(Bear, BaldEagle, True)~~
~~Finish implied with rule7~~
Unbind $x
Bind $x to 'Lion'
Obtain a new implied fact: Chases('Lion', 'Bear', True)
~~Reuse rule4: Chases($x, BaldEagle, True) → Likes($x, Bear, True)~~
~~Bind $x to 'Lion'~~
~~Finish implied with rule4~~
~~Reuse rule7: Visits(Lion, Squirrel, True) ∧ Chases(Squirrel, Bear, True) → Chases(Bear, BaldEagle, True)~~
~~Finish implied with rule7~~
Unbind $x
Bind $x to 'Squirrel'
Obtain a new implied fact: Chases('Squirrel', 'Bear', True)
~~Reuse rule4: Chases($x, BaldEagle, True) → Likes($x, Bear, True)~~
~~Bind $x to 'Squirrel'~~
~~Finish implied with rule4~~
~~Reuse rule7: Visits(Lion, Squirrel, True) ∧ Chases(Squirrel, Bear, True) → Chases(Bear, BaldEagle, True)~~
~~Finish implied with rule7~~
Unbind $x
Finish implied with rule9

Use rule7: Visits(Lion, Squirrel, True) ∧ Chases(Squirrel, Bear, True) → Chases(Bear, BaldEagle, True)
Obtain a new implied fact: Chases('Bear', 'BaldEagle', True)
Finish implied with rule7

Finally, we obtain following implied facts:
Likes(', 'Bear', True)
Chases('BaldEag'Squirrelle', 'Bear', True)
Chases('Lion', 'Bear', True)
Chases('Squirrel', 'Bear', True)
Chases('Bear', 'BaldEagle', True)
Young('Squirrel', False)

Finish reasoning

Turn-3 (from human):
Question: Based on the above information, is the following comment true, false, or unknown? The bear chases the bald eagle.

Options:
A) True
B) False
C) Unknown

Now you should:
1. reason first carefully with the above known facts and implied facts
2. then output your answer strictly in the format of "So, the correct option is: {X}". The {X} should be replaced by "A) True", "B) False" or "C) Unknown"

Turn-4 (from LM):
The comment can be parsed into: Chases('Bear', 'BaldEagle', True)
~~According to the known fact: Chases('Bear', 'Lion', True)~~
~~The comment contradicts the fact.~~
~~So, the correct option is: B) False~~
According to the implied fact: Chases('Bear', 'BaldEagle', True)
The comment is consistent with the fact.
So, the correct option is: A) True

Figure 7: Another error case from the test set of ProofWriter where LoGiPT (CodeLlama-13b-hf) fails to backtrack in Turn-2 accurately and omits some necessary reasoning steps, leading to missing essential reference Fact in Turn-4. The incorrect reasoning steps are marked in yellow, and the correct revisions or missing steps are marked in green.