Harnessing the Power of Large Language Models for Natural Language to First-Order Logic Translation

Yuan Yang[1], Siheng Xiong[1], Ali Payani[2], Ehsan Shareghi[3] & Faramarz Fekri[1]
[1]Georgia Institute of Technology, [2]Cisco, [3]Monash University
{yyang754@,sxiong45@,faramarz.fekri@ece.}gatech.edu
apayani@cisco.com ehsan.shareghi@monash.edu

Abstract

Advancements in logical reasoning, utilizing LLMs to convert natural language into logical symbolism, combined with the use of external theorem provers, have repositioned the symbolic approach as a central point of interest. The main challenge within this paradigm lies in the LLMs' capability to accurately translate natural language (NL) statements into first-order-logic (FOL) expressions. Although LLMs have shown notable success, there remains a gap in understanding the limitations and challenges they encounter in NL-FOL translation. This is primarily due to the absence of datasets and evaluation test beds at the required fine-grained level. We present MALLS, a dataset of 28K diverse and verified sentence-level NL-FOL pairs collected from GPT4. We utilize a combined strategy of FOL rule parsing, human annotation, and automatic filtering to ensure quality. We also present LOGICLLAMA, a LLaMA2-7B/13B fine-tuned on MALLS for NL-FOL translation, which can be used standalone or to correct previously generated rules by GPT3.5 after being further fine-tuned via a novel reinforcement learning with human feedback (RLHF) framework. We benchmark a wide range of LLMs on MALLS and previous datasets, highlighting weaknesses in them in NL-FOL translation and demonstrating the advantages of MALLS. We also show that LOGICLLAMA achieves GPT4-level performance and can generalize to other datasets. Project repo is available here.

1 Introduction

The traditional logic-based approaches to reasoning (Bos and Markert, 2005; Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010), once regarded as foundational for many core NLP tasks, lost popularity due to their limited scalability and coverage. The rise of end-to-end deep neural networks has garnered significant attention because of their impressive downstream performance (Selsam et al., 2018; Wang et al., 2021), although at the cost of losing interpretability. The recent remarkable progress of large language models (LLMs) has revitalized interest in logic, placing it once again at the forefront of reasoning. One straightforward approach to harness the power of LLMs such as GPT4 (OpenAI, 2023) is to first translate natural language (NL) statements (e.g., premises and conclusions in textual entailment task) to first-order logic (FOL) formula via in-context learning, and then pass the symbolic forms to Symbolic Mathematical Theory (SMT) solvers. This approach has significantly surpassed the performance of LLM-only methods such as Chain-of-Thought (Wei et al., 2022) and has established new state-of-the-art for complex deductive reasoning tasks (Pan et al., 2023; Ye et al., 2023; Olausson et al., 2023).

Despite the success of this new promising direction, very little is understood about the difficulties current LLMs face in the NL-FOL translation task itself. This is mainly due to the absence of fine-grained curated data of NL statements and their corresponding FOL, with controlled complexity and diversity that enables a categorical study of the capabilities and shortcomings of LLMs in translation. This is crucial to understand what exact kinds of complexity the model builders should target to improve. Annotating the FOL rules for NL statements is expensive and requires domain expertise. Most of the existing logic reasoning datasets do not have parallel FOL annotations. LogicNLI (Tian et al., 2021) and FOLIO (Han et al., 2022) are the few exceptions that provide NL-FOL pairs with sentence-level FOL annotation (Table 1). However, LogicNLI pairs are synthetically generated from a few templates, which do not reflect real-world commonsense, whereas FOLIO pairs suffer from NL-FL misalignment and are too small for sufficient fine-tuning.

Dataset	Source	#NL-FOL pairs	NL			FOL								
			Vocab size	Avg. #words	Mean Perplexity	Avg. #literals	$\forall$	$\exists$	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\oplus$
LogicNLI[3]	Synthetic	12K	2061	13.9	215.0	2.8	2783	5327	10230	6590	2373	8712	3288	0
FOLIO[3]	Expert	2K	5105	10.4	203.2	2.1	1111	182	421	631	167	1137	17	121
MALLS-test	GPT4+Expert	1K	20959	15.7	51.7	4.6	27256	1738	3443	25209	5164	25212	1835	1777
MALLS-dev	GPT4+Filter	27K												

	Examples
LogicNLI	**NL:** If someone is either not wet or not every, then he is not jealous and not sharp. **FOL:** $\exists x(\neg \texttt{wet}(x) \vee \neg \texttt{every}(x)) \rightarrow (\neg \texttt{jealous}(x) \wedge \neg \texttt{sharp}(x))$. **Error:** synthetic statement; does not reflect commonsense
	NL: All guilty people are not tall. **FOL:** $\exists x \texttt{guilty}(x) \rightarrow \neg \texttt{tall}(x)$. **Error:** synthetic statement; does not reflect commonsense
FOLIO	**NL:** Zaha Hadid is a British-Iraqi architect, artist and designer. **FOL:** $\texttt{British-IraqiArchitect}(zahaHadid)$. **Error:** oversimplified FOL; does not align with NL
	NL: If people order takeout from delivery services often, then they grow their own fresh vegetables in their home garden. **FOL:** $\forall x (\texttt{Takeout}(x) \rightarrow \texttt{Garden}(x))$. **Error:** oversimplified FOL; does not align with NL
MALLS	**NL:** A car must have a motor and wheels to be considered functional. **FOL:** $\forall x\ \texttt{Car}(x) \wedge \texttt{Functional}(x) \rightarrow (\texttt{HasMotor}(x) \wedge \texttt{HasWheels}(x)))$. **Error:** none
	NL: A grocery store sells food and household items. **FOL:** $\forall x \exists y \exists z (\texttt{GroceryStore}(x) \wedge \texttt{Food}(y) \wedge \texttt{HouseholdItem}(z) \wedge \texttt{Sells}(x,y) \wedge (\texttt{Sells}(x,z))$. **Error:** none

Table 1: Statistics of LogicNLI, FOLIO, and MALLS. MALLS consists of the largest set of verified NL-FOL pairs that are more diverse and natural than prior works.

We present MALLS (large language **M**odel gener**A**ted **NL**-FOL pair**S**), a 28K real-world, diverse, and verified sentence-level NL-FOL pairs collected from GPT4. Compared to prior datasets (Table 1), MALLS **enjoys better diversity in terms of vocabulary, context, and complexity and is the largest dataset to date in this category**, which provides sufficient samples for evaluation and fine-tuning. To create MALLS, we implement an adaptive pipeline that prompts GPT4 for pairs with diverse context. Knowing that GPT4 can generate misaligned FOL rules, we take extra caution in verifying the generated pairs. We verify the dataset with human annotation, identify categorical mistakes made by the GPT4, and filter the dataset from 34k to 28k. We further analyze the type of errors GPT4 made, revealing categorical areas of weakness in the latest LLMs.

On top of MALLS, we present LOGICLLAMA, a LLaMA2-7B/13B model family (Touvron et al., 2023) for NL-FOL translation fine-tuned with LoRA (Hu et al., 2021). LOGICLLAMA can be used for (1) directly translating NL to FOL as a standalone translator and (2) can be used in combination with more powerful general-purpose models such as GPT3.5, where it serves a "correction model" that corrects outputs from GPT3.5, which we found yields better performance with a fraction of the cost of GPT4 [1]. In particular, we propose a novel data augmentation and reinforcement learning with human feedback (RLHF) framework that trains LOGICLLAMA on the synthetically perturbed NL-FOL pairs using a FOL verifier as the reward model, making LOGICLLAMA to achieve GPT4-level performance.

We summarize our contributions as follows: (1) We present MALLS, a 28K real-world, diverse, and verified sentence-level NL-FOL pairs collected from GPT4, which, to the best of our knowledge, is the largest dataset to date in this category. (2) We also present LOGICLLAMA, a LLaMA2-7B/13B model family fine-tuned on MALLS, which, in the experiments, achieves GPT4 level performance on NL-FOL translation tasks on three benchmarks.

2 Related Work

NL-FOL translation. NL-FOL translation has been a long-standing challenge in both NLP and the formal logic literature. It is a critical task and plays a central role in many real-world logic-based AI systems and applications, such as natural language theorem proving (Abzianidze, 2017), logic-grounded natural language inference (NLI) (Bos and Markert, 2005; Han et al., 2022; Pan et al., 2023), logic-grounded QA (Wang et al., 2021), controlled text generation (Poesia et al., 2023; Lu et al., 2020). The lack of a high-performing translation model with wide data coverage is the main obstacle for these systems to generalize to more tasks.

Traditionally, NL-FOL translation is addressed via rule-based methods (Abzianidze, 2017; Zettlemoyer and Collins, 2005; Bos and Markert, 2005).

[1]As of Feb 2024, GPT3.5 costs \$0.002/1K tokens for completion whereas GPT4 costs \$0.06/1K.

Due to the complexity of natural language, these methods are difficult to scale to real-world applications. Recently, there has been an increasing interest in approaching this task via neural approaches (Lu et al., 2022; Cao et al., 2019; Hahn et al., 2022; Wang et al., 2021; Singh et al., 2020; Levkovskyi and Li, 2021), which gives rise to a new paradigm of using LLMs for translation.

Recent works (Pan et al., 2023; Ye et al., 2023; Olausson et al., 2023) have successfully used GPT models for translation via in-context learning (ICL). While they demonstrate impressive downstream task performance, it is crucial to study how these LLMs perform in fine-grained categories. More importantly, GPT-based ICL methods come with several limitations: (1) As of Feb 2024, GPT-4 costs \$0.06/1K tokens and can be expensive to run; GPT-3.5, while costs less, struggles with complex translation (§5); (2) they are closed-source and subject to constant updates, making it difficult for privacy-sensitive use cases and reproducing results for academic purposes; (3) they lack extendability for future integration with downstream tasks such as NLI and QA. Therefore, is it valuable to have small and open-source LLMs with GPT-level performance while preserving privacy and extendability.

NL-FOL datasets. Most of the existing logic reasoning datasets such as LogiQA (Liu et al., 2020), RuleTaker (Clark et al., 2020), ReClor (Yu et al., 2020) and text2log (Levkovskyi and Li, 2021), either do not provide sentence-level FOL annotations, or the annotations are generated without verification. Among them, LogicNLI (Tian et al., 2021) and FOLIO (Han et al., 2022) are closest to our work, which provides NL statements with parallel FOL annotations. However, pairs in LogicNLI are generated synthetically and share a similar FOL template. FOLIO consists of real-world expert-written pairs, but the size of 2K is insufficient for fine-tuning an LLM. This work extends the prior work and proposes to collect NL-FOL pairs from GPT4. As a result, MALLS has collected 28K pairs that are more diverse in terms of context and complexity. In experiments, we evaluate LOGICLLAMA on MALLS as well as on LogicNLI and FOLIO and demonstrate that MALLS is of high quality and enables models to generalize to other benchmarks.

3 MALLS

Creating a dataset of diverse NL statements with faithfully aligned FOL annotations is crucial to evaluating and fine-tuning LLMs for logic-based reasoning. Existing datasets (Table 1) are either synthetic or too small for such purposes. To this end, we create the MALLS dataset by collecting NL-FOL pairs from GPT4 which is considered to be the most powerful LLM to date.

3.1 Prompt pipeline

To collect data from GPT4, we implemented a prompt pipeline that dynamically adjusts the prompts to ensure the *diversity* and *validity* of the NL-FOL pairs. The pipeline consists of the following modules: (1) **N-gram frequency counter**; (2) **Prompter**; and (3) **FOL rule verifier**.

N-gram frequency counter. During prompting, we keep track of the frequencies of the N-grams in the entire NL statement corpus. Specifically, we track 1- and 3-grams. Once the frequency of a specific N-gram in the collected data reaches the frequency threshold (500 and 250 respectively), we will instruct GPT4 to not produce any NL-FOL pairs including it. For example, "... *DO NOT involve concepts and terms (and the synonyms) such as animal, food, ...*". The list of N-grams in the instruction grows as more reach the frequency threshold.

Prompter. A prompter assembles the prompts generated from different modules (prompt table shown in §A): (1) SYSTEM PROMPT: specifying the basic requirements such as the syntax and generation format. (2) FEW-SHOT EXAMPLES PROMPT: consisting 5 NL-FOL pair examples randomly sampled from the corpus. Initially, pairs are sampled from the FOLIO dataset and later on from the GPT4-generated ones (we checked to ensure none of the FOLIO examples, or close variations are leaked into the GPT4 generated NL-FOL pairs.). This diversifies the prompts and leads to less similar examples. (3) NEGATIVE N-GRAM PROMPT: instructing GPT4 not to involve frequent N-grams (introduced earlier) in the generated NL-FOL pairs. (4) FOL PROMPTS: generating prompts that specify the desired form of FOL rules, i.e., the number of variables and whether or not to include more logic operators such as $\oplus$, $\neg$, and $\vee$ which we found GPT4 tends to ignore in default generation. These configurations are picked randomly every

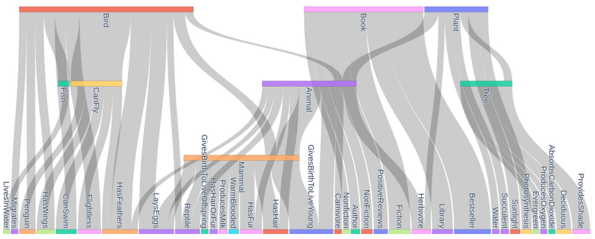


Figure 1: Snippet from the top 200 frequent FOL term pairs in MALLS (for full version see Appendix A). Many terms are associated with a wide range of other terms, suggesting the rules are semantically diverse.

time the prompt is generated. (5) BREAK-DOWN PROMPT: We found GPT4 by default tends to make over-complicated predicates that absorb important logical meanings. For example, " ### NL: *A fruit is considered ripe if it is mature and its color has changed from green to red.* ### FOL: $\forall x(\texttt{Fruit}(x) \wedge \texttt{Mature}(x) \wedge \texttt{ColorChangedToRed}(x) \rightarrow \texttt{Ripe}(x))$. " The predicate ColorChangedToRed is complicated and should be broken down into "$\texttt{ColorBefore}(x, y) \wedge \texttt{ColorAfter}(x, z) \wedge \texttt{Green}(y) \wedge \texttt{Red}(z)$". We detect long predicate names and include a prompt encouraging the model to break down the rules.

FOL rule verifier. GPT4 can sometimes generate syntactically invalid FOL rules. We implement a verifier that checks the syntax of the rules. Specifically, we specify the context-free grammar (CFG) of the expected FOL rule and parse the generated FOL with NLTK [2] CFG parser, and erase those that could not be parsed (grammar and example parse trees in §A).

3.2 NL-FOL alignment check

Apart from the syntax validity we also ensure FOL rules align with the NL sentences. The straightforward approach is to check each pair manually as that in FOLIO (Han et al., 2022). However, conducting a full annotation is prohibitive for the size of MALLS (x14 times larger) for an academic budget. Instead, we employ a hybrid approach where we first manually annotate a subset of MALLS, identify the common error modes, and then implement a filtering module to filter those in the entire set. We find this approach to be effective and increase the correct ratio from 84.5% to 92.8%.

MALLS **Annotation**. We annotate a subset of 1K examples uniformly sampled from MALLS. Each sample is rated with respect to a 3-point

[2]https://www.nltk.org/

Free Var. (32.7%)	Nested Equiv. (30.7%)
NL: Children who are vaccinated against a disease have a reduced risk of contracting that disease. **FOL:** $\forall x \forall y(\texttt{Child}(x) \wedge \texttt{VaccinatedAgainst}(x, y) \rightarrow \texttt{ReducedRisk}(x, y))$ **Error:** undefined free variable y	**NL:** A liquid is considered boiling if its temperature is at or above its boiling point. **FOL:** $\forall x(\texttt{Liquid}(x) \rightarrow (\texttt{Boiling}(x) \leftrightarrow \texttt{TemperatureAtOrAboveBoilingPoint}(x)))$ **Error:** wrong usage of $\leftrightarrow$
Nested Impl. (18.0%)	Aggressive Xor (1.9%)
NL: Drinking a lot of water and getting enough sleep is good for your health. **FOL:** $\forall x(\texttt{Person}(x) \rightarrow ((\texttt{DrinksALotOfWater}(x) \wedge \texttt{GetsEnoughSleep}(x)) \rightarrow \texttt{GoodForHealth}(x)))$ **Error:** unnecessarily nested Impl.; redundant predicate Person	**NL:** A guitar has strings, while a keyboard has keys. **FOL:** $\forall x \forall y(\texttt{Guitar}(x) \wedge \texttt{Keyboard}(y) \rightarrow (\texttt{HasStrings}(x) \oplus \texttt{HasKeys}(y)))$ **Error:** NL does not imply that HasStrings contradicts with HasKeys
Missing NL Info (15.1%)	Other (1.6%)
NL: Ice cream melts when exposed to heat and becomes solid when frozen. **FOL:** $\forall x(\texttt{IceCream}(x) \rightarrow (\texttt{ExposedToHeat}(x) \oplus \texttt{Frozen}(x)))$ **Error:** "becomes solid" is missing in FOL	**NL:** An athlete can be a professional if they are paid for their performance. **FOL:** $\forall x(\texttt{Athlete}(x) \wedge \texttt{PaidForPerformance}(x) \rightarrow \texttt{Professional}(x))$ **Error:** quantifier should be $\exists$

Table 2: Examples of different error types. The percentage (%) reported on each column denotes the share of each specific error within the total errors observed.

scale: correct, partial, and incorrect. Each sample is evaluated by 3 annotators and gets the final rating via majority votes. We recruited 6 graduate students in CS/ECE department with a background in FOL. The annotators were provided with detailed instructions regarding the rating scales, and several examples per rating to clarify further (details in §A). The annotators were given 3 hours to finish the rating of 500 examples. The labeled 1K subset consists of 84.5% correct, 11.8% partial, and 3.7% incorrect pairs. We measure Krippendorff's alpha for the inter-annotator agreement rate alignment check, which is 0.714, indicating the annotations are generally aligned.

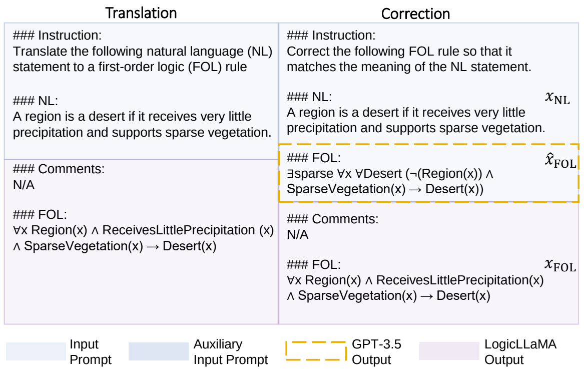
Filtering MALLS. We analyze the 1K samples, identify four common error modes (Table 2), and design corresponding modules to filter them: (1) Free variables. GPT4 occasionally generates FOL rules with free variables such as "$\forall x P(x) \wedge R(x, y)$", where y is not bounded. We remove them by parsing the rule and checking if it contains free variables. (2) Nested equivalence/implication. For some complex NL statements such as "*If an A has B, C, D then it is E*", GPT4 tends to confuse between "$\wedge$", "$\rightarrow$", and "$\leftrightarrow$" and generates inaccurate rules such as "$\forall x A(x) \rightarrow (B(x) \wedge C(x) \wedge D(x) \leftrightarrow E(x))$". We find samples with this nested structure all have a similar alignment issue, so we remove these pairs by parsing the rule and checking if it contains such a structure. (3) Aggressive xor. We find GPT4 struggles with the meaning of "$\oplus$", and may generate a rule such as "$A(x) \oplus B(x)$" for an NL sentence "*A or B*". To remove those pairs with aggressive xor usage, we construct a 5-shot prompt and use GPT3.5 to identify the potential pairs and then remove the incorrect ones manually. (4) Missing NL information.

We find GPT4 may omit important entities or concepts in the FOL rule. We adopt a similar treatment as (3), where we first do 5-shot in-context learning with GPT3.5 to identify potentially incorrect samples and then remove the true incorrect ones manually. After removing these incorrect samples, the resulting 1K samples have 92.8% correct, 6.0% partial, and 1.2% incorrect pairs. We filter the raw MALLS of 34K samples the same way and obtain the final 28K verified samples.

3.3 MALLS statistics

General statistics. We show the general statistics in Table 1 together with those of LogicNLI and FOLIO[3]. MALLS contains 28K NL-FOL pairs, which is significantly larger than LogicNLI and FOLIO, and different from LogicNLI which is synthetically generated, the pairs are also more diverse and contextually rich, where the NL statements have a vocabulary size of 20.9K and an average length of 15.7 compared to 10 in FOLIO. By leveraging the strong natural language compatibility of GPT4, the NL statements of MALLS are more natural and better in reflecting real-world commonsense, reaching 1/4 of the mean perplexity (computed with GPT2) than LogicNLI and FOLIO. For FOL rules, the average number of literals reached 4.6 indicating more complex rules (also see Figure 8 in §A).

Pair diversity. The NL-FOL rules in MALLS are highly diverse. To see this, we investigate the frequencies and the correlations of the FOL *terms*. A *term* is either a predicate name or a named entity in a FOL rule. For example, "$\forall x((\texttt{Person}(x) \land \texttt{Drinks}(x)) \rightarrow \texttt{DependentOn}(x, \texttt{Caffeine}))$" consists of 4 terms, i.e., `Person`, `Drinks`, `DependentOn` and `Caffeine`. MALLS has a total term vocabulary size of 49394 and the most frequent terms occur less than 2K times (Figure 9 in §A), suggesting a diverse vocabulary distribution. On the other hand, we investigate the correlations between terms and illustrate the top 200 frequent term pairs. We show a snippet of this in Figure 1 (full diagram at Figure 7 in §A). Note that if a term is associated with many other terms,



```
### Instruction:
Translate the following natural language (NL)
statement to a first-order logic (FOL) rule

### NL:
A region is a desert if it receives very little
precipitation and supports sparse vegetation.

### Comments:
N/A

### FOL:
∀x Region(x) ∧ ReceivesLittlePrecipitation (x)
∧ SparseVegetation(x) → Desert(x)
```

```
### Instruction:
Correct the following FOL rule so that it
matches the meaning of the NL statement.

### NL:
A region is a desert if it receives very little
precipitation and supports sparse vegetation.

### FOL:
∃sparse ∀x ∀Desert (¬(Region(x)) ∧
SparseVegetation(x) → Desert(x))

### Comments:
N/A

### FOL:
∀x Region(x) ∧ ReceivesLittlePrecipitation(x)
∧ SparseVegetation(x) → Desert(x)
```

Figure 2: Input and expected outputs for translation and correction.

this typically means the rules involving that term are diverse in semantics and context, and Figure 1 suggests that it is indeed the case. For example, for rules involving `Book`, they cover the knowledge of its genre (e.g., `Fiction`), places (e.g., `Library`), viewership (e.g., `Bestseller`), and so on.

4 LOGICLLAMA NL-FOL Translation

Besides evaluation, MALLS makes it possible to fine-tune local LLMs such as LLaMA2-7B/13B to reach GPT4-level performance, which we refer to as LOGICLLAMA. Unlike typical NLP tasks, where one fine-tunes with a naive autoregressive objective, fine-tuning for NL-FOL translation is nontrivial. In §4.1 and §4.2, we present three ways to train LOGICLLAMA. And in §4.3, we propose two metrics for evaluating an FOL rule against the ground truth.

4.1 Autoregressive fine-tuning

The LOGICLLAMA can be trained to directly translate the FOL from NL, i.e., **(T1) translation** task; it can also be trained to *correct* the generated FOL from a more powerful model such as GPT3.5, i.e., **(T2) correction** task. Intuitively, we found in experiments that GPT3.5 is good at doing the "heavy-lifting" part of the translation and can capture the main part of the FOL rule; then presumably, one can train a smaller model that corrects the output from the GPT3.5 to get a better result.

Figure 2 shows the input and output sequence of the two tasks: let $\langle \boldsymbol{x}_{\text{NL}}, \boldsymbol{x}_{\text{FOL}} \rangle$ be an NL-FOL pair from MALLS; for **(T1)**, the input and output are the original sequences $\boldsymbol{x}_{\text{NL}}$ and $\boldsymbol{x}_{\text{FOL}}$ respectively; and for **(T2)**, let $\hat{\boldsymbol{x}}_{\text{FOL}} = \text{GPT}(\boldsymbol{x}_{\text{NL}})$ be the FOL predicted by GPT3.5, the input is the NL and the prediction put together $[\boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}}]$ and the output is the ground-truth FOL, $\boldsymbol{x}_{\text{FOL}}$. We train both

[3]Note that the FOLIO statistics are different from those reported in (Han et al., 2022). As of 2023, the released dataset misses the ground truth FOL annotations for conclusions in the training set, and some pairs contain duplicates and invalid FOL rules. We removed those during pre-processing. Also, the LogicNLI statistics are obtained from the official repo here, which contains 12K samples instead of the 20K reported in the paper.

(T1) and **(T2)** via standard autoregressive objective. Specifically, we fine-tune LLaMA2-7B/13B with LoRA for all the attention and feedforward linear layers on MALLS.

4.2 RLHF fine-tuning

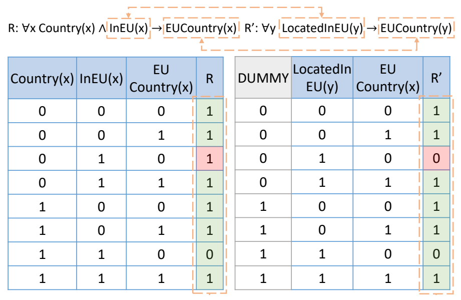
While **(T1)** translation and **(T2)** correction are easy to train, they do not lead to optimal performance. This is due to FOL rules having multiple logically equivalent forms. For example, "$A \leftarrow B$" is equivalent to "$A \vee \neg B$", and an LLM trained with text-level autoregression is not capable of recognizing the equivalent forms. To enable LOGICLLAMA to comprehend the FOL rules, we propose to fine-tune LOGICLLAMA via RLHF with a logical equivalence solver (discussed in §4.3) as the reward model. To do so, as shown in Figure 12, we augment the MALLS with synthetic FOL rules by perturbing the ground-truth and GPT3.5-generated FOL rules and train the model to correct the perturbed rules. We refer to this mode as **(T3) RLHF Correction**.

FOL Rule Perturbation. In **(T2)** correction, the model is trained to correct samples generated by GPT3.5. These samples are mostly correct but they lack diversity in covering various FOL forms and levels of correctness. Considering the high variance of RLHF training, such a dataset would not be sufficient. To this end, we propose to generate synthetic FOL rules by randomly perturbing the FOL rules in MALLS. We leave details of perturbation to §B. We generate 150K perturbed samples in the form of $\langle \boldsymbol{x}_{\text{NL}}, \boldsymbol{x}_{\text{FOL}}, \hat{\boldsymbol{x}}_{\text{FOL}} \rangle$, where $\hat{\boldsymbol{x}}_{\text{FOL}}$ is the FOL rule perturbed from $\boldsymbol{x}_{\text{FOL}}$.

RLHF Correction. To conduct RLHF fine-tuning, a reward model that captures logical equivalence between FOL rules is needed, which compares the final corrected rule to the ground-truth rule and measures how close they are. This gives rise to an RL approach to the problem. Formally, let $\texttt{RM} : \mathcal{X} \times \mathcal{X} \mapsto [0, 1]$ be a function that maps a pair of FOL sequences, $\boldsymbol{x}_{\text{FOL}}$ and $\boldsymbol{x}'_{\text{FOL}}$, to a scalar score representing the pair similarity, our objective can be formalized as maximizing the score (effectively the expected return in RL),

$$\max_{\pi} \texttt{RM}(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}}), \text{where} \qquad (1)$$
$$\boldsymbol{x}'_{\text{FOL}} \sim \pi_{\theta}(\boldsymbol{x}_{\text{FOL}} | \boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}}),$$

for all tuples $\langle \boldsymbol{x}_{\text{NL}}, \boldsymbol{x}_{\text{FOL}}, \hat{\boldsymbol{x}}_{\text{FOL}} \rangle$ in the augmented dataset via a policy $\pi_{\theta}(\boldsymbol{x}_{\text{FOL}} | \boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}})$ which is exactly the base autoregressive model we would


R: ∀x Country(x) ∧ InEU(x) → EUCountry(x)
R': ∀y LocatedInEU(y) → EUCountry(y)

Country(x)	InEU(x)	EU Country(x)	R
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

DUMMY	LocatedIn EU(y)	EU Country(x)	R'
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Figure 3: Example of computing the Logical Equivalence score, 7/8=0.875.

like to fine-tune in **(T3)**. With objective Eq.(1), task **(T3)** is now similar to the RLHF [4] proposed in InstructGPT (Ouyang et al., 2022) with the only difference being the reward model RM, where in our case, RM is a logical equivalence solver (§4.3) instead of a language model and the model is optimized via PPO (Schulman et al., 2017). We summarize key aspects of three tasks in §B and training settings in §D.

4.3 FOL evaluation

Consider two FOL rules "$R : \neg(P(A) \wedge P(B))$" and "$R' : \neg P(A) \vee \neg P(B)$" — R and R' are logically equivalent but are different in raw text; also consider a pair of rules "$R : \forall x P(x)$" and "$R' : \forall x \forall y P(x) \wedge Q(y)$" — if R is the ground-truth and R' the LLM prediction, how should one measure the distance between the two?

Logical equivalence (LE). We propose to measure the logical equivalence between the rules by matching their truth tables and computing the overlap ratio. We introduce this with a running example in Figure 3. Specifically, let R and R' be the two rules parsed from the text $\boldsymbol{x}_{\text{FOL}}$ and $\boldsymbol{x}'_{\text{FOL}}$. We identify the set of literals in each rule $\mathcal{P} = [p_1, p_2, ...]$ and $\mathcal{Q} = [q_1, q_2, ...]$. For Figure 3, $\mathcal{P} = [\texttt{Country}(x), \texttt{InEU}(x), \texttt{EUCountry}(x)]$ and $\mathcal{Q} = [\texttt{LocatedInEU(y)}, \texttt{EUCountry}(y)]$. We consider the set of literals as an array of Boolean variables, and the FOL as a circuit that takes in the Boolean values and outputs a single Boolean value. Therefore, we can represent a FOL with a truth table that enumerates all inputs and the resulting outputs. To compare R and R', we count the number of configurations that match and divide it by the total number of configurations; this yields a score in $[0, 1]$. In Figure 3, this is $7/8 = 0.875$. The main issue with this approach

[4]Technically, (T3) does not involve human feedback, but we keep the name since the protocols are the same.

is finding the right input bindings between $\mathcal{P}$ and $\mathcal{Q}$, and dealing with the case where the numbers of inputs are different (i.e., $|\mathcal{P}| \neq |\mathcal{Q}|$). We solve this by finding the binding that gives the highest LE score via greedy search and filling the rest of the missing inputs with dummy inputs. In Figure 3, $\texttt{InEU}(x)$ binds to `LocatedInEU(y)` and $\texttt{EUCountry}(x)$ binds to $\texttt{EUCountry}(y)$; and we fill in a dummy in $\mathcal{Q}$ to match $\texttt{Country}(x)$ in $\mathcal{P}$. We leave more details in §C.

Reward design for RLHF correction. The reward model RM in **(T3)** measures the similarity between two text FOLs $\boldsymbol{x}_{\text{FOL}}$, $\boldsymbol{x}'_{\text{FOL}}$. Therefore, we use the LE score as the main source of the reward. However, we also want the model to extract the right predicate and entity names from the NL statement. We incorporate this aspect by computing the BLEU score between the text $\boldsymbol{x}_{\text{FOL}}$ and $\boldsymbol{x}'_{\text{FOL}}$ with a specialized FOL tokenizer. We set the final reward as the mixture of the two: $\text{RM}(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}}) = \omega * \text{LE}(R, R') + (1-\omega) * \text{BLEU}(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}})$, where ω is the mixing ratio and in experiments we set it to 0.7. With this setting, we encourage the model to prioritize on generating the logical equivalent rules rather than ones that are similar in plain text with a high BLEU score.

5 Experiments

We address the following questions in the experiment section: (**Q1**) How good is MALLS for evaluating existing LLMs? Does it provide better insights into the shortcomings? (**Q2**) How good is MALLS for fine-tuning? Can we train LOGICLLAMA that also generalizes to other datasets such as FOLIO? (**Q3**) How well does the LOGICLLAMA perform in direct translation mode and correction mode?

Dataset. For MALLS, we use the human-annotated subset (1K) as the test set; we hold out 2K samples as the valid set, and the rest samples are used for training **(T1-3)**; we also include 1K pairs from the training set of LogicNLI since it has a different rule distribution where rules are mostly grounded rules (i.e., many of them do not contain any variables) instead of FOL rules. We evaluate the LLMs on MALLS test set as well as the full FOLIO dataset and the test set of LogicNLI.

Methods. To answer (**Q1**) and (**Q2**), we evaluate LOGICLLAMA together with a wide range of LLMs on the task of NL-FOL translation: LLaMA2-7B/13B, Mistral-7B, Mixtral, Gemini

Methods	LogicNLI		FOLIO		MALLS	
	FOL BLEU	FOL LE	FOL BLEU	FOL LE	FOL BLEU	FOL LE
LogicLLaMA-7B Trans.	0.912	0.965	0.354	0.826	0.762	0.910
LogicLLaMA-7B Corre.	0.913	0.970	0.368	0.827	0.767	0.910
LogicLLaMA-7B RLHF Corre.	0.923	0.979	0.378	0.841	0.774	0.920
LogicLLaMA-13B Trans.	0.922	0.978	0.361	0.832	0.765	0.916
LogicLLaMA-13B Corre.	0.925	0.982	0.377	0.852	0.769	0.918
LogicLLaMA-13B RLHF Corre.	0.927	0.986	0.384	0.858	0.778	0.924

Table 3: BLEU and LE scores of different LOGICLLAMA variants. Corrections are performed over GPT3.5 outputs.

Pro, GPT3.5, and GPT4. Apart from LOGICLLAMA, all LLMs are tested with the in-context learning (ICL) setting with 0-shot, 5-shot, and 10-shot examples (we skip 0-shot for LLaMA models because they are not instruction-fine-tuned). We also include a Chain-of-Thought (CoT) setting: the LLMs are given three CoT examples, which decompose the translation into several steps, such as identifying the predicates, identifying the entities and variables, and forming the final FOL rules.

Metrics. We use the logical equivalence (LE) introduced §4.3 as the main metric for FOL rules evaluation. Apart from this, it is also preferred that the generated FOL rule has the matching predicate names, so we measure the FOL BLEU between the predicted FOL and the ground-truth FOL. Note that BLEU is a noisy metric: BLEU is computed over the FOL text tokens and does not recognize the underlying logic structures, thus can have a low score even if the rules are equivalent.

5.1 Results

Figure 4 shows the results of LOGICLLAMA (i.e., LOGICLLAMA-13B RLHF Corre.) and other LLMs on LogicNLI, FOLIO, and MALLS. Concerning (**Q1**), a good translation benchmark should provide a consistent and challenging evaluation of the method. That said, for LogicNLI, most methods achieve near-perfect results as one increases the ICL shots. This is because LogicNLI is synthetically generated and the rules all share a similar FOL template, making it a poor benchmark for this task. On the other hand, FOLIO is more challenging. The LE metric indicates the FOL rules are more diverse. However, the BLEU metric is significantly worse than its LE counterpart and other benchmarks. After careful inspection, we found this is largely due to the NL-FOL misalignment issue in FOLIO dataset, where FOL rules omit details in the NL sentence and use highly simplified predicate names (as shown in Table 1). Therefore, we conclude that FOLIO is still lacking as a translation benchmark. In compari-

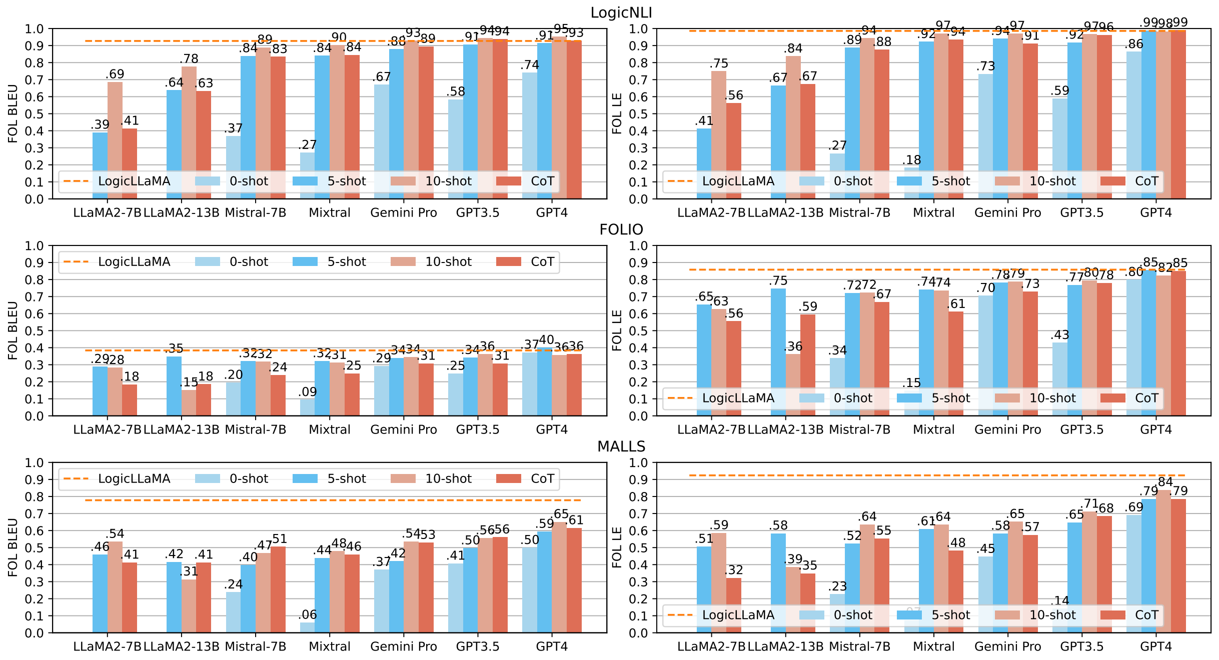


Figure 4: BLEU and LE scores of LLMs. Here, LOGICLLAMA reports the results of LOGICLLAMA-13B RLHF Correction over GPT3.5 output.

son, MALLS is more balanced in terms of BLEU and LE metrics and is generally most challenging. We observe that simply increasing ICL shots has diminishing returns and the best ICL method besides GPT4 (since it is the model that generates MALLS) can only achieve 0.71 LE score. This means MALLS is sufficiently diverse and challenging, and cannot be solved with simple ICL settings—This answers the question (**Q1**).

LOGICLLAMA **results**. As shown in Figure 4, LOGICLLAMA outperforms most the ICL methods and reaches GPT4-level performance for three benchmarks. Note that LOGICLLAMA is fine-tuned on MALLS and generalizes well to FOLIO dataset. This indicates MALLS is a high-quality dataset for fine-tuning, addressing question (**Q2**).

Analysis of LOGICLLAMA **variants**. Table 3 shows the performance of all LOGICLLAMA variants. We found that translation, correction, and RLHF correction lead to increasing performance, and the 13B one generally outperforms the 7B one on all three benchmarks. This confirms our intuition in §4.1 and addresses the question (**Q3**). More importantly, it suggests a new paradigm of future LLM development: by training a local LLM on the output of a more powerful model, one can customize the model behavior while still leveraging the powerful ones for heavy lifting.

Performance analysis. We further analyze how the methods perform across different level sample complexities and logic categories. Figure 5 shows the LE scores of LOGICLLAMA-13B RLHF Corre. and other LLMs (5-shot ICL) over four different splits of MALLS-test: # NL tokens, NL perplexity, # FOL tokens, and # FOL literals. In general, the larger these values, the more complex the samples are, thus more challenging for translation. Figure 5 suggests that all ICL methods' performance decreases significantly as the samples become difficult. Remarkably, we find LOGICLLAMA is more robust to the increase of sample complexity, indicating that MALLS-dev is a well-balanced set and the fine-tuning on it is effective. On the other hand, Figure 6 shows the LE scores over FOL rules consisting of different types of logic operators. We find LOGICLLAMA consistently outperforms other baselines over all categories. We also observe that baseline LLMs, especially the open-source ones, tend to perform worse on operators such as $\leftrightarrow$ and $\oplus$. We conjecture that this is because such FOL rules are rarer in the pretraining corpus as they are not frequently used in practice and can be subsumed by conjunction normal forms (CNF). This also highlights the importance of creating open-source datasets such as MALLS, which enables insight into such issues.

6 Conclusion

We release MALLS, a high-quality dataset of 28K verified sentence-level NL-FOL pairs collected

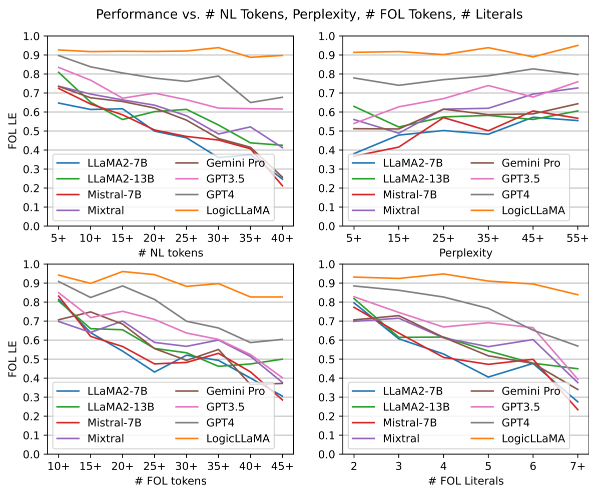


Figure 5: Performance vs 4 types of MALLS-test splits.

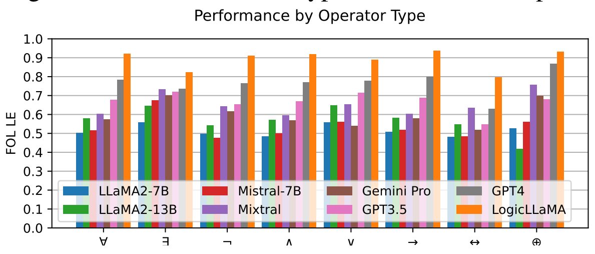


Figure 6: Performance by logic operator types.

from GPT4. During the verification process, we identify patterns of errors made by GPT4. This underscores the difficulty in achieving accurate NL-FOL translation, warranting further research. We also present LOGICLLAMA, the first specialized LLMs for the NL-FOL translation task fine-tuned on MALLS. In experiments, LOGICLLAMA shows competitive performance with GPT4 on three NL-FOL benchmarks.

7 Limitations

During the creation of MALLS, we used a combined approach of human annotation and auto filtering to create MALLS-test and MALLS-dev respectively. While unlikely, it is possible that MALLS-dev still contains unaligned, inaccurate NL-FOL pairs as they are not individually inspected by a human annotator.

Also, note that MALLS contains sentence-level NL-FOL pairs, where the logical rules are in first-order. This choice has several implications: (1) NL-FOL pairs are generated independently and they do not share a common predicate name space, variable space, or entity space. This means one cannot use MALLS to train a reasoner to translate a multi-sentence story into a unified logic system with shared predicates and entities as those in FOLIO. (2) FOL rules are encoded with first-order logic syntax, which is different from other formats such as SMT-lib. The model trained on MALLS generate rules only with an FOL format. Nonetheless, our source code comes with a text parser that parses raw text FOL rule into a CFG tree, which makes further conversion to other formats straightforward. (3) The semantics of NL-FOL pairs are generated by GPT4, meaning it may or may not reflect factual knowledge. We create MALLS to evaluate and train LLMs for the NL-FOL translation task, which could be done in real-world context where NL statements are all factual, or in hypothetical scenario where NL statements are made up. Therefore, we deliberately skip the factual check on the generated pairs and focus on the alignment between NL and FOL. This means the pairs may contain information that is hypothetical or not factual.

8 Ethics Statement

Potential negative impact. Since the NL-FOL pairs are not created to be always factual, training on MALLS might introduce misinformation into an LLM. Also, when used in the intended manner, it is likely the trained models, such as LOGICLLAMA, still produce incorrect FOL rules, which potentially leads to wrong answers in the downstream reasoning task.

Artifact statements. We release data, code, and weights under Apache 2.0 license and they are intended for research use only. Additionally, the usage of the LOGICLLAMA model should follow the license agreement of LLaMA and Alpaca. MALLS is released under CC BY NC 4.0 and the use of such dataset should follow the policy of OpenAI, as it is collected from GPT-4.

Human annotation. To create MALLS-test, We invited 6 graduate students in CS/ECE department with a background in FOL. The annotators were provided with detailed instructions regarding the rating scales, and several examples per rating to clarify further (details in §A). The annotators were given 3 hours to finish the rating of 500 examples. The task is approved by the IRB board of the institute. The participants are given instructions and asked for consent to participate in this study at the beginning of the task and they can opt out at any time. The final dataset MALLS-test only contains the aggregated label for each sample and no information about the identity, demographic, and geographic characteristics of the participants is collected, or stored during the process.

References

Lasha Abzianidze. 2017. LangPro: Natural language theorem prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 115–120, Copenhagen, Denmark. Association for Computational Linguistics.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 51–64, Florence, Italy. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.

Christopher Hahn, Frederik Schmitt, Julia J Tillman, Niklas Metzger, Julian Siber, and Bernd Finkbeiner. 2022. Formal specifications from natural language. *arXiv preprint arXiv:2206.01962*.

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 Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. FOLIO: Natural Language Reasoning with First-Order Logic. ArXiv:2209.00840 [cs].

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233.

Oleksii Levkovskyi and Wei Li. 2021. Generating predicate logic expressions from natural language. In *SoutheastCon 2021*, pages 1–8. IEEE.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. *arXiv preprint arXiv:2010.12884*.

Xuantao Lu, Jingping Liu, Zhouhong Gu, Hanwen Tong, Chenhao Xie, Junyang Huang, Yanghua Xiao, and Wenguang Wang. 2022. Parsing natural language into propositional and first-order logic with dual reinforcement learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5419–5431, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5153–5176. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, 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*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. 2018. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*.

Hrituraj Singh, Milan Aggrawal, and Balaji Krishnamurthy. 2020. Exploring neural models for parsing natural language into first-order logic. *arXiv preprint arXiv:2002.06544*.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2021. Logic-driven context extension and data augmentation for logical reasoning of text. *arXiv preprint arXiv:2105.03659*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satisfiability-aided language models using declarative prompting. *CoRR*, abs/2305.09656.

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

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

A MALLS Dataset Creation Details

A.1 Data collection

Prompt table. Table 4 shows the prompt templates used for prompt generation.

A.2 FOL parsing and verification

FOL CFG grammar. We define the FOL with the following CFG grammar:

```
S -> F | Q F
Q -> QUANT VAR | QUANT VAR Q
F -> '¬' '(' F ')' | '(' F ')' | F OP F | L
OP -> '⊕' | '∨' | '∧' | '→' | '↔'
L -> '¬' PRED '(' TERMS ')' | PRED '(' TERMS ')'
TERMS -> TERM | TERM ',' TERMS
TERM -> CONST | VAR
QUANT -> '∀' | '∃'
```

Note that, for PRED, CONST, and VAR they have corresponding production rules generated for each FOL rule example. For example, for rule "$\forall x((\texttt{Person}(x) \land \texttt{Drinks}(x)) \rightarrow \texttt{DependentOn}(x, \texttt{Caffeine}))$", the production rules are

$$\texttt{PRED} \rightarrow \texttt{"Person"} \mid \texttt{"Drinks"} \mid \texttt{"DependentOn"}$$
$$\texttt{CONST} \rightarrow \texttt{"Caffeine"}$$
$$\texttt{VAR} \rightarrow \texttt{"x"}.$$

We show two example parse trees in Figure 10 and Figure 11.

A.3 NL-FOL alignment check

We invite 6 graduate students to label 1K samples, where each sample is annotated by 3 annotators on a 3-point scale: correct ,partial, and incorrect. We provide detailed instruction as follows:

You will evaluate how well the FOL rule aligns with the meaning of the NL sentence on a 3-point scale: Correct, Partial, and Incorrect. Let's use a running example to show you how to rate the translation:

"""

NL: `A turtle has a shell and can swim.`

FOL: $\exists x(\texttt{Turtle}(x) \land \texttt{Shell}(x) \land \texttt{CanSwim}(x))$

"""

Your evaluation involves checking the following rubrics:

- Whether the rule has the right predicates, i.e., "Turtle", "Shell", and "CanSwim".
  - Having a slightly different predicate name is fine (e.g., "Shell" instead of "hasShell"); this is still a Correct example.
  - However, if some important notions in the NL are missing in the FOL (e.g., "Shell" missing in the FOL) or some predicates in FOL are not present in the NL (e.g., "CanRun(x)" found in FOL), it is one Partial error.
- Whether the rule has the right quantifiers.
  - You should only examine this item if the NL explicitly contains the following phrases: If NL explicitly says "Every A has sth" or "For all A..." then the FOL must have "∀". On the other hand, if it says "Some A has sth" or "There exists A that...", then the FOL must have "∃". Failing to have the right quantifiers is one Partial error.
  - If NL does not have these explicit hints, then using either "∀" or "∃" is fine (e.g., the turtle example above can use either "∀" or "∃")
- Whether the FOL rule has the same logical meaning as the NL sentence. This involves comprehending the meaning FOL rule and comparing it against the NL sentence
  - FOL misaligns with the NL meaning due to a wrong expression being used. It is one Partial error if the FOL simply misaligns with but not contradicts the NL; but if the meaning explicitly contradicts the NL, then it is an Incorrect example. E.g.:
    - Using the wrong logic operators, e.g., "Turtle(x) ∨ Shell(x) " instead of "Turtle(x) ∧ Shell(x)", is an Incorrect example as the meaning is obviously different.
    - Having the wrong implication, e.g., "A person has legs" translated into "HasLegs(x) → Person(X)", where it should be "Person(X) → HasLegs(x)". This is also an Incorrect example.
  - Caveat on "→" vs "∧" vs "↔": Sometimes these three operators can be used interchangeably to express the same meaning despite they are logically inequivalent. For example,

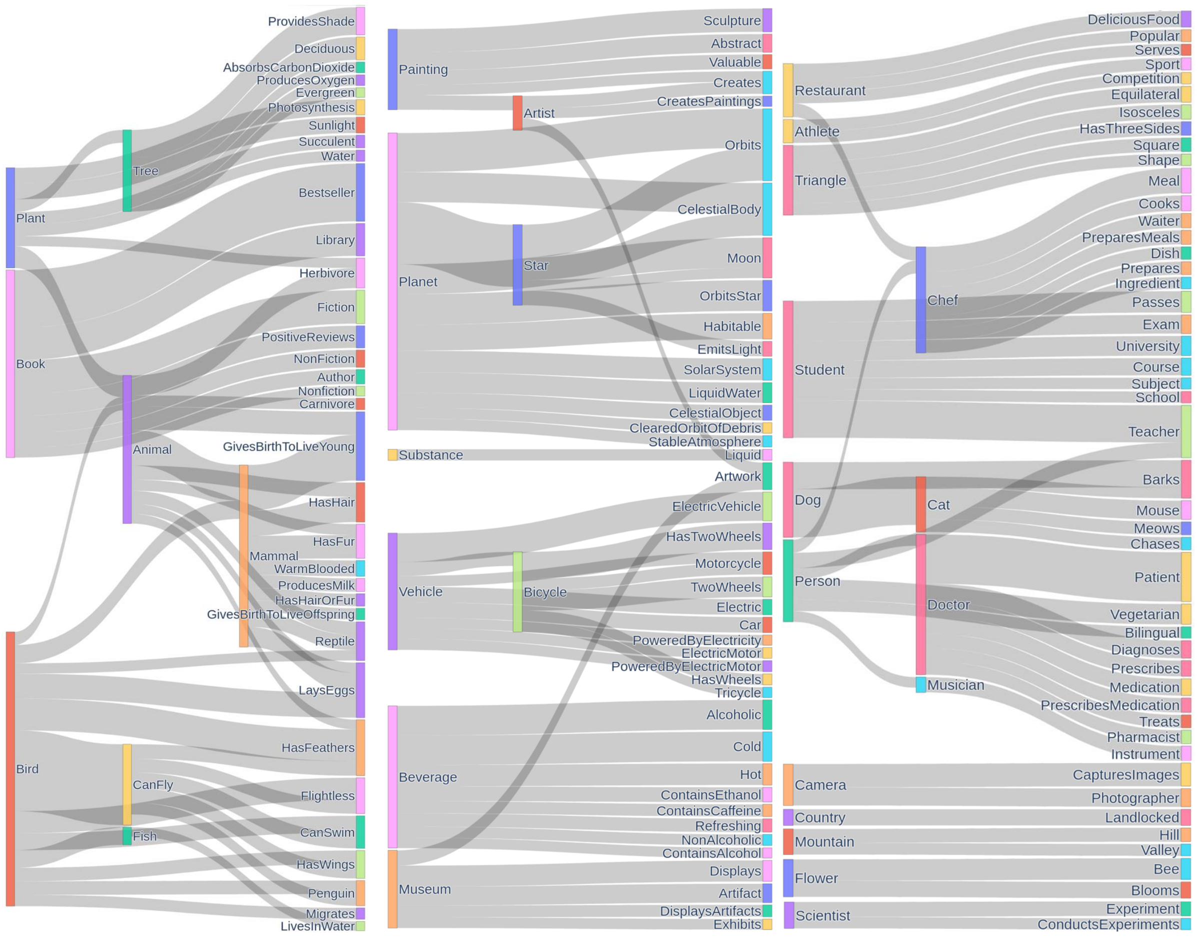


Figure 7: Top 200 frequent FOL term pairs in MALLS. Many terms are associated with a wide range of other terms, which suggests the rules are semantically and contextually diverse.

  * "Turtle(x) → Shell(x) ∧ CanSwim(x)", "Turtle(x) ∧ Shell(x) ∧ CanSwim(x)", and "Turtle(x) ↔ Shell(x) ∧ CanSwim(x)" all express a similar meaning, and should be considered a Correct case.
  * However, there are cases where they obviously lead to different meanings and this should be considered one Partial error.

– Caveat on "∨" vs "⊕": we know "A ∨ B" includes the case that A and B are both True, whereas, "A ⊕ B" means either A or B is True but not both. So how to judge which one is better in the alignment?

  * If NL explicitly says "either A or B but not both" or something similar, then FOL must use "⊕", otherwise using "∨" is fine even if it does not reflect the reality. E.g., "water is solid or liquid" then it is fine to have "Water(x) ∨ Solid(x) ∨ liquid(x)" even though we know it cannot be in two states at the same time.
  * If NL does not explicitly say the above, but FOL contains "⊕", then we need to judge with common sense. For example, "Water(x) → Solid(x) ⊕ liquid(x)" also aligns with the above despite no explicit hints in NL.

– Logic variable issues:

  * Associating the wrong logic variables, e.g.: "if x is greater than y and y is greater than z, then x is greater than z" translated to "GreaterThan(x, y) ∧ GreaterThan(y, z) → GreaterThan(z, x)", where it should be "GreaterThan(x, y) ∧ GreaterThan(y, z) → GreaterThan(x, z)". This is one Partial error.

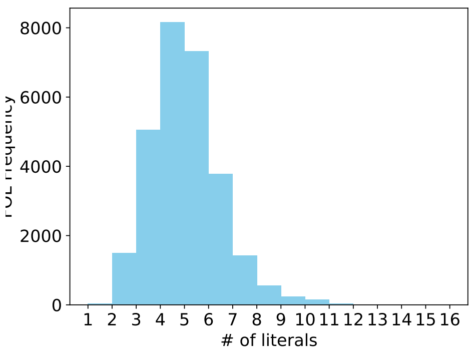


Figure 8: Literal frequency distribution (MALLS).

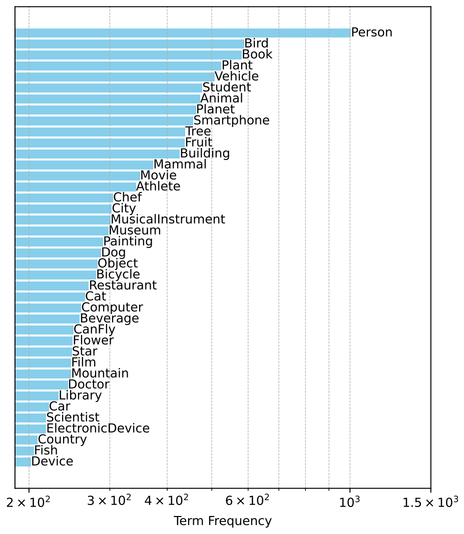


Figure 9: Top 40 frequent FOL terms (MALLS).

- ∗ Redefining variables, e.g. "Turtle(x) ∧ Fish(x) ∧ CanSwim(x)", where variable x is defined with two unary predicates (or classes) where one does not subsume the other one. This is one Partial error. For cases like "Animal(x) ∧ Turtle(x)", it is a Correct use because Animal subsumes Turtle
- ∗ Free/redundant logic variables, e.g.:
  - · "Person(x) ∧ Has(x, y)" — It has a free variable/redundant variable y, which is one Partial error.
  - · "Person(x) ∧ Has(x, y) → Ownby(y, x)" — Note that this is a Correct use of variables: variables do not need to be defined by unary predicates (or "classes") to be valid, e.g., "Tool(y)", as long it appears at least twice in different literals.

By checking the above rubrics, you will rate the rule with one of the four ratings:

- Correct:
  - The FOL rule aligns with the meaning of the NL sentence
  - Possible slightly different predicate names or quantifiers.
  - Some ambiguities in the usage of "→" vs "∧" vs "↔", but it does not lead to an obviously different meaning
- Partial: one and only one of the following errors occur
  - The FOL rule does not align with the NL in some parts but does not explicitly contradict the NL
  - Missing important predicates, or having obviously irrelevant predicates
  - Wrong quantifier when the NL has explicit hints such as "For all…" or "Some…"
  - Wrong use of variables, such as free/redundant variables, wrong variable association, or redefining variables
- Incorrect:
  - More than one Partial error was found in an example
  - The FOL rule explicitly contradicts the NL meaning.

A.4 MALLS statistics

General statistics. Figure 9 and Figure 8 show the top 40 frequent FOL terms and the literal count distribution in MALLS.

Frequent FOL term pairs. Figure 7 shows the top 200 frequent FOL term pairs in MALLS.

B Details on RLHF Correction

B.1 FOL Rule Perturbations

We generate synthetic FOL rules by randomly perturbing the FOL rules in MALLS. We consider three types of atomic perturbations: *label change*, *insert*, and *delete*. As shown in Table 5, label change can be conducted on any terms or logic operators in a FOL rule; insert operation is applicable to term, negation, and formula; and delete operation can be considered as the inverse of insertion. Note that, we restrict the perturbations to only produce valid rules. The reasons are twofold: (1) the invalid rule space is effectively the space of all possible strings which is prohibitive to explore; and (2) we found GPT3.5 rarely generates syntactically invalid rule, thus, limiting the synthetic data in the valid rule space will already cover a wide range of the actual GPT3.5 outputs.

Operation Type	Subtypes	Original	Perturbed
Label Change	Change Predicate	$P(A) \land R(B)$	$R(A) \land R(B)$
	Change Term	$\forall x \;\; P(x) \land P(B)$	$\forall y \;\; P(x) \land P(B)$ $\forall x \;\; P(x) \land P(x)$
	Change Operator	$\forall x \;\; P(x) \land P(B)$	$\forall x \;\; P(x) \lor P(B)$
Insert	Insert Term	$\forall x \;\; P(x) \land P(B)$	$\forall x \; \exists y \;\; P(x) \land P(B)$ $\forall x \;\; P(x) \land P(x, B)$
	Insert Negation	$P(A) \land P(B) \land P(C)$	$P(A) \land \neg(P(B) \land P(C))$
	Insert Formula	$P(A) \land P(B)$	$P(A) \land P(B) \rightarrow R(C)$
Delete	Delete Term	$\forall x \, \forall y \;\; P(x) \land R(x, y)$ $\forall x \, \forall y \;\; P(x) \land R(x, y)$	$\forall y \;\; P(x) \land R(x, y)$ $\forall x \, \forall y \;\; P(x) \land R(y)$
	Delete Negation	$\neg(P(A) \land P(B))$	$\neg(P(A) \land P(B))$
	Delete Formula	$P(A) \land P(B) \;\land P(C)$	$P(A) \land P(C)$

Table 5: The list of all atomic perturbations.

Perturbation process. Given a ground-truth pair $\langle \boldsymbol{x}_{\text{NL}}, \boldsymbol{x}_{\text{FOL}} \rangle$, a parser (§A) will parse $\boldsymbol{x}_{\text{FOL}}$ into an abstract syntax tree (AST). We randomly perturb the AST with atomic operations in Table 5 and for N_{Perturb} times. Here, N_{Perturb} is also picked randomly from a list of numbers, and in the experiments, we set it to $\{0, 1, 2, ..., 10\}$. In the case $N_{\text{Perturb}} = 0$, the perturbed rule remains the same as the ground truth and the step is simply "No changes needed"; this is effectively a negative example that penalizes the model for over-correcting. During training, we found LogicLLaMA still tends to over-correct the samples as negative samples by default account for around 10% of the data, so we manually set the probability of negative sample generation to 0.2.

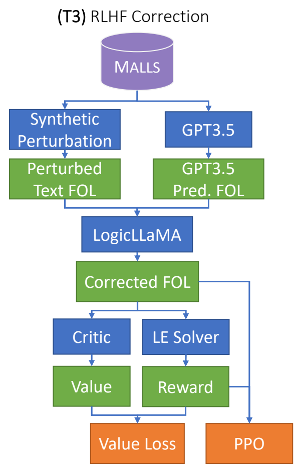


Figure 12: Overview of the training for **(T3)** RLHF correction.

Data augmentation. For RLHF training, we generate the synthetic dataset consisting of 150K examples in the form of $\langle \boldsymbol{x}_{\text{NL}}, \boldsymbol{x}_{\text{FOL}}, \hat{\boldsymbol{x}}_{\text{FOL}} \rangle$ using the above method. We then fine-tune the LLaMA-7B/13B with LoRA again using the standard autoregression objective: the input is $[\boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}}]$ and the output is $\boldsymbol{x}_{\text{FOL}}$. The training process is also illustrated in Figure 12

B.2 Summary of training tasks

We summarize the input, output, and training objectives of **(T1-3)** in Table 6. For task **(T1)** and **(T2)**, a standard autoregressive loss is used: where the model is trained to predict the token x_t given the previous tokens $\boldsymbol{x}_{<t}$ with cross-entropy loss

$$L(\theta) = \mathbb{E}_{x,t}[-\log p_\theta(x_{t,\text{out}} | \boldsymbol{x}_{<t,\text{out}}, \boldsymbol{x}_{\text{in}})], \quad (2)$$

where the $\boldsymbol{x}_{\text{in}}$ and $\boldsymbol{x}_{\text{out}}$ are shown in 6.

Task	Input	Output	Objective
(T1) Translation	$\boldsymbol{x}_{\text{NL}}$	$\boldsymbol{x}_{\text{FOL}}$	Eq.(2)
(T2) Correction	$[\boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}}]$	$\boldsymbol{x}_{\text{FOL}}$	Eq.(2)
(T4) RLHF Correction	$[\boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}}]$	$\boldsymbol{x}_{\text{FOL}}$	Eq.(3)

Table 6: Summary of input, output and objectives of task **(T1-T4)**

For task **(T3)**, the RLHF objective is used: we implement a reward model $\mathtt{RM}(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}})$ that measures the logical equivalence between two FOL rules and train the model to maximize the reward with the PPO algorithm (Schulman et al., 2017). Specifically, the full reward is computed as

$$\begin{aligned} R(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}}) &= \mathtt{RM}(\boldsymbol{x}_{\text{FOL}}, \boldsymbol{x}'_{\text{FOL}}) - \\ \beta * \log & \frac{\pi_\theta(\boldsymbol{x}_{\text{FOL}} | \boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}})}{\pi^{\text{ref}}(\boldsymbol{x}_{\text{FOL}} | \boldsymbol{x}_{\text{NL}}, \hat{\boldsymbol{x}}_{\text{FOL}})}, \end{aligned} \quad (3)$$

where the policy π_θ is the language model f_θ with learnable parameters θ and π^{ref} is the reference model from **(T3)** with parameters frozen. The second term in Eq.(3) is the KL-divergence between the learned policy and its original policy from **(T3)** and its strength is controlled by a hyperparameter β.

C Computing Logical Equivalence and BLEU Score

logical equivalence. To train and evaluate LogicLLaMA, we compute the logical equivalence score (LE) that measures the similarity between two rules R and R'. The computation is done in three steps: (1) finding the literals of R and R', that is $\mathcal{P} = [p_1, p_2, ...]$ and $\mathcal{Q} = [q_1, q_2, ...]$; (2) binding the literals in $\mathcal{P}$ to those in $\mathcal{Q}$ (or vice versa); and (3) generating the truth tables for the binding and computing the score.

Finding the literals of a FOL rule is straightforward after we parse it into a CFG tree: we extract all the subtrees whose root label is `L` and remove possible duplicate literals. In the case where the parsing fails, we simply skip the rest of the computation and return a score of zero, as that indicates the rule is syntactically invalid.

The main challenge here is to determine the literal binding between $\mathcal{P}$ and $\mathcal{Q}$. Using Figure 3 as the example, R has literals $\mathcal{P} = [\texttt{Country}(x), \texttt{InEU}(x), \texttt{EUCountry}(x)]$ and R' has literals $\mathcal{Q} = [\texttt{LocatedInEU(y)}, \texttt{EUCountry}(y)]$. We want to find the one-one matching for each of the literals, such that we can compare the truth tables. First, we address the case where $|\mathcal{P}| \neq |\mathcal{Q}|$ by adding `DUMMY` inputs to the shorter one, and in this example, it is $\mathcal{Q}$ which becomes $[\texttt{LocatedInEU(y)}, \texttt{EUCountry}(y), \texttt{DUMMY1}]$. To match the literals, we first determine the matching strategy. Note that there are in total $!|\mathcal{Q}|$ numbers of bindings (permute $\mathcal{Q}$ when keeping $\mathcal{P}$) and there are many strategies to measure the match: for example, one can enumerate all bindings and compute the "average" score of all bindings or finding the worst case of the binding. Here, we choose to find the binding that yields the highest LE score, that is the "best" case binding. To do this, we implement a simplistic greedy search algorithm that iterates over each literal in $\mathcal{P}$ and finds the closest literal in $\mathcal{Q}$ in terms of edit distance. To avoid exponential numbers of bindings, we limit the search depth to 1000. Finally, given a binding between $\mathcal{P}$ and $\mathcal{Q}$, we compute the LE score by comparing the rows in their truth tables as the one shown in Figure 3.

FOL BLEU score. We use a specialized tokenizer for computing the FOL BLEU score. This tokenizer splits every quantifier, operator, and term into tokens. The split token sequence is the same as the leave nodes in the CFG parse tree (Figure 10 and Figure 11) listed in pre-order.

D Experimental Settings

For all training tasks, we fine-tune LogiCLLaMA using LoRA with rank=8, $\alpha = 8$, and dropout 0.05 on all the linear layers in decoder blocks "[q_proj,k_proj,v_proj,o_proj, gate_proj,down_proj,up_proj]". We use the AdamW optimizer (Loshchilov and Hutter, 2017) with $lr = 0.0003$. For the generation, we use a cutoff length of 256 For **(T1)** and **(T2)**, the generation uses temperature=0.1, top_p=0.75, top_k=40 and num_beams=1. For **(T3)**, we adopt the setting suggested in the TRL library, which uses top_k = 0.0, top_p = 1.0, do_sample = True and no eos token; this effectively lets the model sample tokens from the logits and always generate to the full length. This generation configuration is needed to compute a valid KL divergence score between the actor model and the reference model (a copy of the same model before training). Task **(T1)** and **(T2)** generally take about 5 GPU hours to complete with a single GTX 4090. For **(T3)**, we train it for 50 GPU hours with a single A100. We report all results from a single experiment run.

Table 4: List of prompt templates used for prompting GPT4 for NL-FOL pairs.

System prompt	I want to create a dataset for translating natural language (NL) statements into first-order logic (FOL) rules. You will help me to create a diverse set of NL-FOL pairs. For natural language (NL) generation, you should: 1. Come up with a statement stating either complex or simple real-world commonsense facts 2. The statements are meaningful, and diverse from each other For FOL rule generation: 1. You SHOULD USE the following logical operators: ⊕ (either or), ∨ (disjunction), ∧ (conjunction), → (implication), ∀ (universal), ∃ (existential), ¬ (negation), ↔ (equivalence) 2. You *SHOULD NEVER USE* the following symbols for FOL: "'", "≠", "%", "=" 3. The literals in FOL SHOULD ALWAYS have predicate and entities, e.g., "Rounded(x, y)" or "City(guilin)"; expressions such as "y = a ∨ y = b" or "a ∧ b ∧ c" are NOT ALLOWED 4. The FOL rule SHOULD ACCURATELY reflect the meaning of the NL statement 5. You SHOULD ALWAYS put quantifiers and variables at the beginning of the FOL 6. You SHOULD generate FOL rules with either: (1) no variables; (2) one variable "x"; (3) two variables "x", "y"; or (4) three variables "x", "y" and "z" Generation Format: you SHOULD ALWAYS generate the NL and FOL pairs in the following format """ — NL: {your generated NL} — — FOL: {your generated FOL} — """
Initial FOLIO Few-shot examples prompt	— NL: If someone is entire, then he is not serious, and vice versa. — FOL: ∃x entire(x) ↔ ¬serious(x) — NL: If there is at least one people who is both not excited and not timid, then Jonathan is elderly. — FOL: ∀x (¬excited(x) ∧ ¬timid(x)) → elderly(Jonathan) — NL: Someone who is eithor not fresh or entire is always not serious. — FOL: ∀x (¬concerned(x) ∨ fresh(x)) → entire(John) — NL: If Nathalie is not blue, then Collier is entire. — FOL: ¬blue(Nathalie) → entire(Collier) — NL: Someone is courteous and not elderly if and only if he is not excited and not various. — FOL: ∃x (courteous(x) ∧ ¬elderly(x)) ↔ (¬excited(x) ∧ ¬various(x))
Negative N-gram prompt	They DO NOT involve concepts and terms (and the synonyms) such as "considered","person","either","water", "if it has","if it is","it has a","is considered a","A person is"
FOL prompts	They are [complex | simple] statements involving at least [1 | 2 | 3] logical variables
	The statement involves diverse logical operators such as logical negation, logical xor and disjunction
Break-down prompt	[IMPORTANT] AVOID making long predicate names like "MoonShinesAtNight","SunShinesDuringDay"

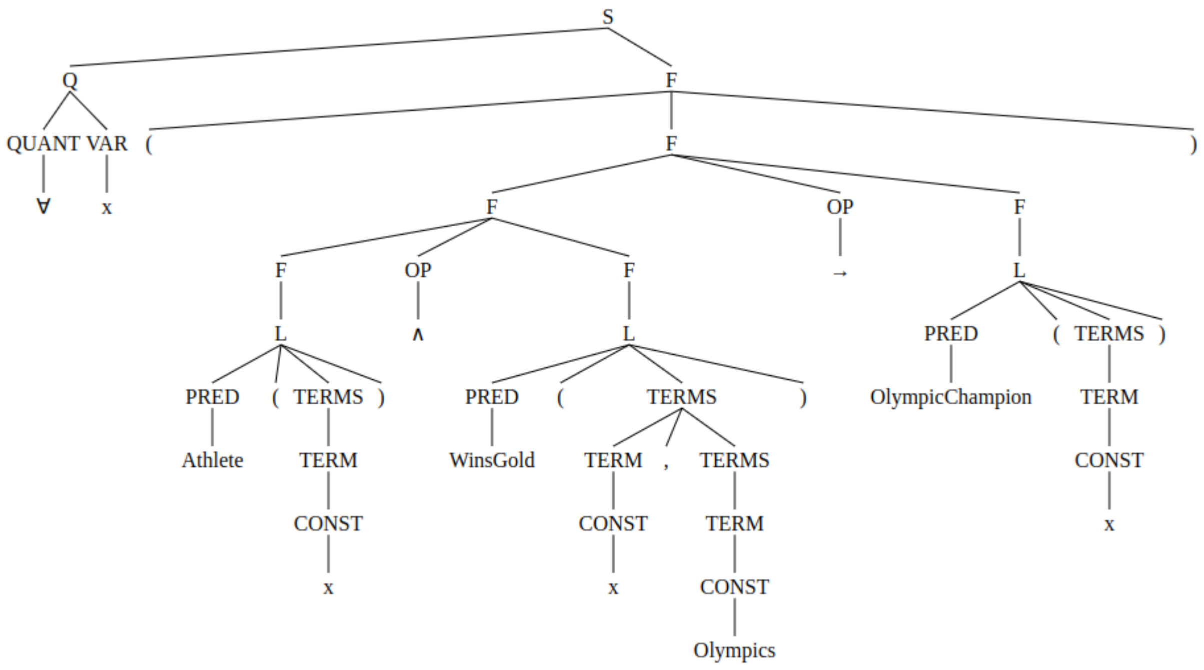


Figure 10: CFG parse tree of FOL rule $\forall x(\texttt{Athlete}(x) \wedge \texttt{WinsGold}(x, \texttt{Olympics}) \rightarrow \texttt{OlympicChampion}(x))$.

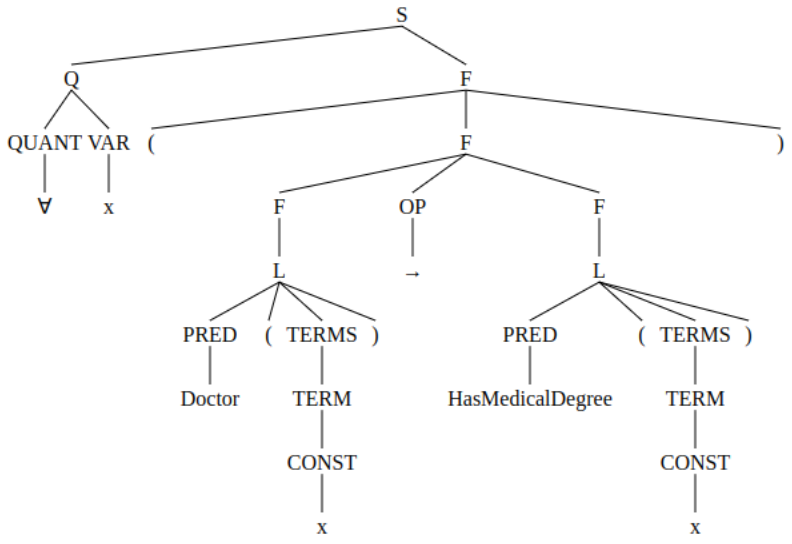


Figure 11: CFG parse tree of FOL rule $\forall x(\texttt{Doctor}(x) \rightarrow \texttt{HasMedicalDegree}(x))$.