Efficient Conversion of NL-based Logical Problems to FOL using LLMs and Preference Optimization

Anonymous ACL submission

Abstract

Logical reasoning is a key task for artificial intelligence due to it's role in major downstream tasks such as Question Answering, Summarization. Recent methods in improving the reasoning ability of LLMs fall short in correctly converting a natural language reasoning problem to an equivalent logical formulation, which hinders the framework's overall ability to reason. Towards this, we propose to use finetuning on a preference optimization dataset to learn to parse and represent a natural language problem as a whole to a consistent logical program by 1) introducing a new supervised and preference optimization dataset (LOGICPO), and 2) adopting popular techniques such as Direct Preference Optimization (DPO), Kahneman-Tversky optimization (KTO) to finetune open-source LLMs. Our best model with PHI-3.5 consistently outperforms GPT-3.5-TURBO's (8-shot) by producing 10% more logically correct and with 14% less syntax errors. Through the framework and our improved evaluation metrics, we offer a promising direction in improving the logical reasoning of LLMs by better representing them in their logical formulations.

1 Introduction

Recent state-of-the-art pipelines for Logical Reasoning tasks show a marked shift from having Large Language Models (LLMs) generate solutions directly. Most frameworks first translate the problem into a formal language using an LLM and then use a corresponding logical engine (such as a theorem prover) to solve reasoning tasks (Pan et al., 2023; Ye et al., 2023; Olausson et al., 2023). For example, Logic-LM (Pan et al., 2023) demonstrates how tasks can be converted into first-order logic, satisfiability (SAT/SMT), or constraint satisfaction problems depending on their types, and solved using publicly available engines. However, the efficiency of these methods are limited due to the reliance on the ability of pretrained LLMs to correctly translate the natural language reasoning problem to a *consistent* logical program. These logical languages are often *low-resource* compared to coding languages such as Python, and therefore (possibly) less prevalent in pretraining data.

In this work, we therefore focus solely on analyzing and then improving the parsing of a natural language-based reasoning *problem* (including a *context* and a *query*) consistently to First Order Logic (FOL) problem. We choose FOL as the target language, as it is a widely adopted logical language with broad expressivity, and one can utilize available algorithms to convert FOL programs to other equivalent languages (SAT, SMT), as necessary. We use two publicly available logical reasoning datasets (with available FOL translations) for benchmarking. First, we analyze the failure modes of open and closed-source LLMs through a comprehensive study (varying prompting strategies). We explore whether utilizing other (more popular) language (such as Python) as an intermediate step and post-process it to generate FOL problems may help. We also observe, contemporary efforts in improving FOL translations primarily target sentence wise translation (Yang et al., 2024), which does not take predicate level consistency into account – *for example, Tab. 1 shows how the last sentence is translated into* `SAT2016(.)` *predicate, that does not match* `SAT(.)` *predicate used elsewhere.*

We observe various syntactic, logical and semantic errors in the FOL programs produced using GPT3.5-TURBO, Llama-3-8B. We use a set of natural language problems (a story S) from FOLIO, consisting of a context, query and a logical label. Utilizing a prompt and varying number of in-context examples, we ask an LLM to generate an multiple output FOL story ($\tilde{FS}$) for an input story. Using Prover9, we obtain the logical label from the generated FOL story ($\tilde{FS}$). Based on whether the logical labels match with the original groundtruth

label or errors from Prover9[1], we create the supervised finetuning and the preference optimization (LOGICPO) dataset. Furthermore, we employ various preference optimization methods (PPO, DPO and KTO) and combine them with open-source large language models to observe how their parsing errors reduce. Lastly, we analyze what errors remain difficult to reduce even after employing our dataset and training methodologies. In summary, our contributions are three-fold:

- First, we evaluate baseline popular open and closed-source LLMs such as Llama, GPT3.5 in their capabilities of generating FOLs from natural language context and query. We extensively benchmark the effect of increasing number of shots, using intermediate representation; and provide an analysis of the errors the models make.
- We introduce a preference optimization dataset (LOGICPO) where the input is textual context and query and the output is an FOL program, capturing various perturbations synthetically to curb errors at a logical, syntactic and semantic level.
- We experiment with various preference optimization algorithms on open-source LLMs such as Llama-3, Gemma-3, Phi-3.5 and Flan-T5. We establish the efficacy of our LOGICPO dataset, in producing consistent FOL program from natural language context and query.

2 Related Work

In NLP, prior to Transformers-based end-to-end systems, the need for Semantic Parsers dominated various downstream applications, such as Question-Answering, and Recognizing Textual Entailment (Gu et al., 2022; Beltagy et al., 2014; Deng et al., 2022). The task of learning to parse involved: 1) converting natural language to expert-defined novel semantic representations (Banarescu et al., 2013; Ge and Mooney, 2005; Sharma et al., 2015; Chanin, 2023) or 2) translating text to sentences in well-established programming languages such as (SQL, First Order Logic). Several solutions around knowledge-based question answering (Kim et al., 2020; Wu et al., 2016; Liu et al., 2024) attempted to convert the natural language question to structured SQL queries (or SPARQL queries) against known schemas. However, such models demanded a large number of parallel data points (source sentence and target parse), and did not explore context-level inter-sentence consistency of such parsing.

In recent times, various researchers have cast doubt on whether LLMs actually reason (Nikankin et al., 2024; Lu et al., 2024). As a remedy, a series of efforts (Olausson et al., 2023; Pan et al., 2023; Ye et al., 2023) have shown that LLMs can be used as few-shot semantic parsers to parse into known programming languages to represent a reasoning problem; and such programs can be executed to find the correct answer to the problem. While LLMs show surprising ability to parse text questions to languages such as Python using in-context learning, their performance for less popular formal languages (such as First-order Logic) are comparatively worse. Researchers resort to multiple tricks, such as self-refinement (Olausson et al., 2023) using error(s) signals from Automated Theorem Provers (such as Z3, Prover9) as additional input. These are quite cost-extensive with multiple LLM calls, and do not greatly improve the conversion accuracy. There have been some efforts using small Language Models (such as T5) (Lu et al., 2022) to learn to parse Natural Language sentences to FOL, most efforts concentrate on benchmarking sentence level parsing, such as LOGICLLAMA (Yang et al., 2024). The authors in LOGICLLAMA utilize GPT-4 sentence-level NL-FOL pairs, followed by filtering to create a data set and use supervised fine-tuning to finetune a Llama model. A recent non-peer reviewed paper introduces ProofFOL (Thatikonda et al., 2024), a high-quality FOL-annotated subset of ProofWriter dataset using GPT-4o. While this dataset would have been somewhat useful, neither the dataset or the trained models are publicly available.

In contrast, our goal is to convert a natural language reasoning problem as a whole to a consistent FOL program. We introduce a preference dataset, perform an extensive study utilzing supervised finetuning and many preference optimization techniques on multiple open-source LLMs. To the best of our knowledge, our work is the first to explore Preference Optimization techniques to train open-source LLMs (such as Llama) to reduce syntactic, and logical errors in FOL parsing. We also plan to release the new preference dataset, that can enable new algorithms and new family of parsers.

[1] https://www.cs.unm.edu/~mccune/prover9/

3 Learning Preference-Optimized NL to FOL Engines

3.1 Natural Language to First-Order Logic Conversion

The FOLIO dataset (Han et al., 2024) is an expert-written dataset containing high-quality examples requiring complex logical reasoning in FOL. The dataset consists of two tasks: natural language reasoning with FOL and NL to FOL translation. We are interested in the second task of NL to FOL translation in this work. The goal of this task is to translate an NL story S to a FOL story FS. The NL story S contains a series of context sentences $p_1, p_2, \ldots p_n$ and a conclusion sentence p_{n+1}. The FOL story FS consists of context formulas $f_1, f_2, \ldots f_n$ and a conclusion sentence f_{n+1}. The translation task requires each p_i in S to be logically and semantically equivalent to its corresponding FOL formula f_i in FS. Moreover, the logical values for the conclusions p_{n+1} and f_{n+1} should be the same based on the corresponding context S and FS. We show an example of the task in Table 1.

3.2 Automated Data Generation

Let's denote X to be an instance of NL to FOL conversion data consisting of an NL story S and a FOL story FS. Let's also consider y to be the logical label of the conclusion sentence/formula constrained on the contextual sentences/formulas. The logical label y always belongs to the set of {*True, False, Uncertain*}.

Setup We first follow the following pipeline to generate a collection of output FOL stories $\bar{FS}$ from given input NL stories S:

- Consider n different examples of X from FOLIO to use as in-context demonstrations.
- Define an appropriate natural language prompted instruction for NL to FOL conversion. Additionally use the n demonstrations in context and ask an LLM to generate the output FOL story $\bar{FS}$ for an input NL story S.
- Consider the logical label of S to be y_{nl}. Predict the logical label $\bar{y}_{fs}$ of the generated story $\bar{FS}$ using a standalone logical engine such as Prover9 (McCune, 2005–2010). Note that the generated logical label $\bar{y}_{fs}$ could be either from {*True, False, Uncertain*} or could also be a {*Error*} in case the generated $\bar{FS}$ cannot compile in the logical engine for any reason.

Dataset for Supervised Fine-Tuning. We construct the supervised fine-tuning (SFT) data by considering instances for which the generated FOL logical label $\bar{y}_{fol}$ matches the given natural language logical label y_{nl}. These SFT instances consist of S as the input and $\bar{FS}$ as the output, as we can consider $\bar{FS}$ to be an equivalent version of S because they both lead to the same logical label.

Dataset for Preference Optimization. Creating preference optimization data requires a preferred and a rejected output sample for the same input. We consider the generated $\bar{FS}$ for which the FOL logical label $\bar{y}_{fol}$ matches the natural language logical label y_{nl} as the preferred sample. On the other hand, $\bar{FS}$ for which the label $\bar{y}_{fol}$ does not match y_{nl} can be considered as the rejected sample. In this case, the preferred and the rejected FOL samples can be considered as the clean and noisy translations of the natural language story, respectively.

Summary of Generated Datasets We denote the SFT dataset as $\mathcal{D}_{sft}$ and the preference dataset as $\mathcal{D}_{pref}$. The summary of the datasets is shown in Table 2.

The input NL stories are drawn from the training set of FOLIO. We used $n = 2, 4, 8$ in-context examples to generate the output FOL stories, as mentioned earlier in Section 3.2. We used the instruction-tuned 8B versions of Llama 3 (Grattafiori et al., 2024), Llama 3.1 (Grattafiori et al., 2024), and the 7B version of Mistral 0.2 (Jiang et al., 2023) as the language models to generate the samples at temperatures of 0.25 and 0.6. We follow this generation setup with multiple LLMs at different temperatures with different in-context LLMs to increase the diversity of the generated outputs. In total, we generate ? output samples for each input NL story in FOLIO. From there, we randomly subsample and select outputs $\bar{FS}$ to create $\mathcal{D}_{sft}$ and $\mathcal{D}_{pref}$, following the same/different logical label matching strategy, as specified in Section 3.2.

3.3 Language Models as NL to FOL Engines

Our objective is to design language models that can convert an NL story S to its corresponding logically consistent FOL story FS. We hypothesize that our automatically generated datasets $\mathcal{D}_{sft}$ and $\mathcal{D}_{pref}$

FOLIO Instance	**Premises**: 1. The SAT test is wholly owned and developed by the College Board. 2. The SAT test is intended to assess student's readiness for college. 3. The SAT was originally designed not to be aligned with high school curricula. 4. Several adjustments were made for the version of the SAT introduced in 2016 to reflect more closely on what students learn in high school. **Premises-FOL**: 1. $Own(sat, collegeBoard) \wedge \neg Own(sat, others)$ 2. $Test(sat, readiness)$ 3. $\forall x(Year(x) \wedge Before2016(x) \Rightarrow \neg AlignHighSchool(x))$ 4. $\exists x(Year(x) \wedge Since2016(x) \wedge AlignHighSchool(x))$ **Conclusion**: Since 2016 the SAT has been better aligned with high school curricula. **Label**: [*True*]
$\mathcal{D}_{sft}$ Output, $\mathcal{D}_{pref}$ Chosen	**Premises-FOL**: 1. $\forall x.(SAT(x) \Rightarrow CollegeBoardOwns(x))$ 2. $\forall x.(SAT(x) \Rightarrow CollegeReady(x))$ 3. $\forall x.(SAT(x) \wedge \neg Aligned(x))$ 4. $\forall x.(SAT(x) \wedge IntroducedIn2016(x) \Rightarrow Aligned(x))$ **Conclusion-FOL**: $\forall x.(IntroducedIn2016(x) \Rightarrow Aligned(x))$ **Label**: [*True*]
$\mathcal{D}_{pref}$ Rejected	**Premises-FOL**: 1. $\forall x.(SAT(x) \Rightarrow CollegeBoard(x))$ 2. $\forall x.(SAT(x) \Rightarrow CollegeReady(x))$ 3. $\forall x.(SAT(x) \wedge \neg AlignedWithHighSchool(x))$ 4. $\forall x.(SAT2016(x) \Rightarrow AlignedWithHighSchool(x))$ **Conclusion-FOL**: $\forall x.(Since2016(x) \wedge AlignedWithHighSchool(x))$ **Label**: [*False*]

Table 1: An example from the FOLIO dataset and its corresponding versions in our $\mathcal{D}_{sft}$ and $\mathcal{D}_{pref}$ datasets. The Label information in $\mathcal{D}_{sft}$ and $\mathcal{D}_{pref}$ is shown for completeness. It is not used during training or inference of our models.

Logical Label	FOLIO	$\mathcal{D}_{sft}$	$\mathcal{D}_{pref}$
True	388	5,001	3,751
False	286	3,169	2,600
Uncertain	330	8,792	3,649
Total	1,004	16,962	10,000

Table 2: Number of instances in FOLIO, $\mathcal{D}_{sft}$ and $\mathcal{D}_{pref}$ across the different logical labels. The logical label correspond to the label of the preferred sample for $\mathcal{D}_{pref}$.

would be useful in teaching language models to become effective NL to FOL conversion engines.

Firstly, $\mathcal{D}_{sft}$ provides parallel NL to FOL conversion data over all the three logical labels of {*True, False, Uncertain*}. This signal would help language models to observe diverse stories with conclusive and inconclusive scenarios. Secondly, $\mathcal{D}_{pref}$ provides signals of what is and isn't the correct FOL conversion of an NL story. The additional signal of how not to convert NL to FOL stories comes from the rejected samples. This would help in teaching language models to avoid issues like syntax errors, which are common in non-fine-tuned models (). The rejected samples also work as hard negatives, as the corresponding FOL story is still closely related to the NL story while being logically inconsistent.

We follow the strategy of i) fine-tuning the language model with $\mathcal{D}_{sft}$ and then ii) fine-tuning with $\mathcal{D}_{pref}$ using commonly used preference optimization algorithms such as DPO and KTO (Ethayarajh et al., 2024). We fine-tune all the parameters of the language model in each of the two stages. We empirically show that this is a useful strategy in making language models highly effective NL to FOL conversion engines. We also analyze the results across various dimensions to find interesting insights (Section 4).

4 Experiments

In this section, we present our experimental setup, datasets, and the baseline models used to compare our parsers. We also give a detailed study of the current parsing abilities of various SOTA LLMs.

4.1 Datasets

Our experiments use tasks from three existing datasets: **FOLIO** (Han et al., 2024), **ProofWriter** (Tafjord et al., 2021) and **PrOntoQA** (Saparov and He, 2023), all of which have been shown to be challenging for off-the-shelf

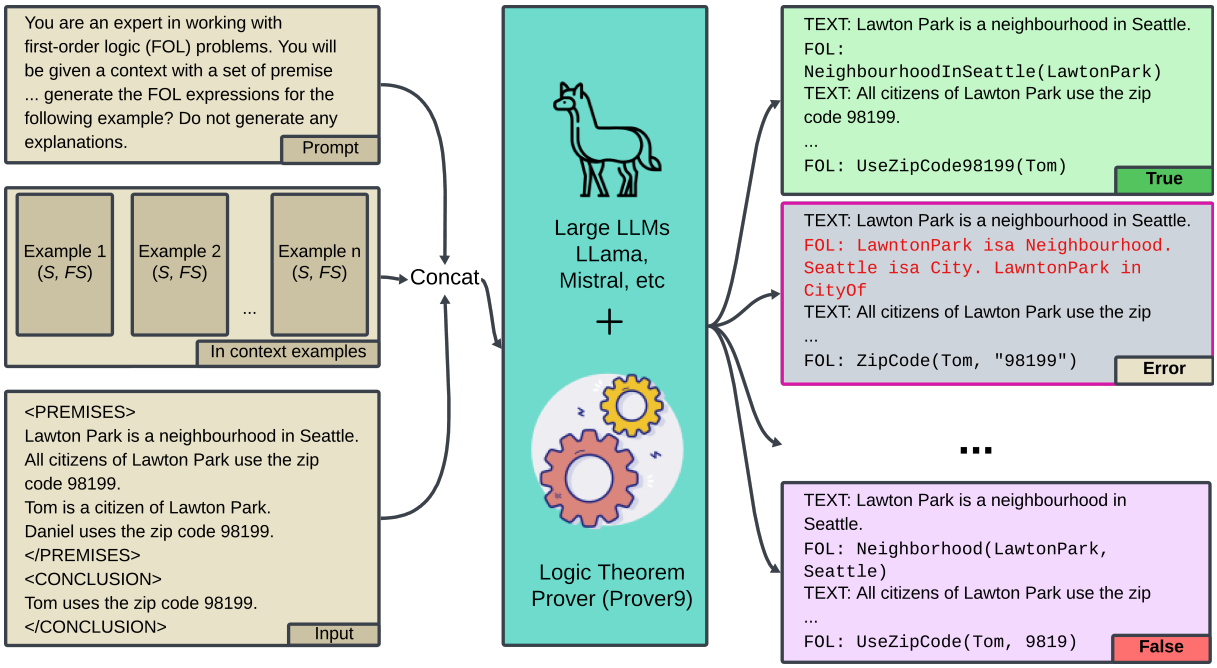

Figure 1: The Data creation pipeline: We use natural language stories from FOLIO, use different settings of LLMs to create various first order logic parses. We then use an off-the-shelf theorem prover (such as Prover9) to observe the predicted logical label of the conclusion. If the predicted label matches the original, we include the sample in $\mathcal{D}_{sft}, \mathcal{D}_{pref}$ (chosen) and if it does not match, we include it as $\mathcal{D}_{pref}$ rejected sample.

LLMs (Olausson et al., 2023). The FOLIO dataset is an expert-written dataset containing high-quality examples requiring complex logical reasoning in FOL. As mentioned in section Section 3.1, we focus on the NL to FOL translation task. We use its validation set for our evaluation, containing 204 examples. Meanwhile, ProofWriter is a synthetically generated dataset for logical reasoning over natural language. We use the sampled split of 360 data points provided by Olausson et al. (2023) for our experiments. The data points are uniformly distributed across the three labels (*True, False, Uncertain*) and maximum question depth (ranging from 0-5; 50 samples each). PrOntoQA is another synthetically generated question-answering dataset, converted from a synthetic world model represented in First Order Logic. It contains triplets of context, query, and a label. We use the validation split that has 500 instances across the (*True, False*) labels. We change the objective of the dataset from question answering to NL-FOL conversion task.

Metrics. We follow (Olausson et al., 2023) to report logical correctness, incorrectness and syntax errors. Deviating from the k-majority voting practice, for each NL story, we generate 10 outputs. For each output FOL story, the Prover9 provides a predicted logical label (or generates syntax error). For correctness, we report the average number of matches to the original label. For incorrectness, we report the average number of times the output label is incorrect. Similar goes for syntax error, the average number of times the output is syntactically incorrect. We additionally report the overall weighted F1 and the F1 score over the True labels.

4.2 Baselines

We compare our approach primarily with two baselines, i.e., the GPT3.5 and GPT4 model variants reported by LINC (Olausson et al., 2023). We refer to them as GPT3.5-LINC and GPT4-LINC respectively. We utilize the outputs for these models provided by (Olausson et al., 2023), to avoid costly experiments of GPT4-LINC. Finally, we evaluate off-the-shelf LLMs in their abilities to generate FOL stories by providing in-context examples.

Few-shot Variants. Prior to fine-tuning, we evaluate the models' parsing abilities by using in-context examples in a few-shot setting. We use the in-context examples from Olausson et al. (2023) in *1, 2, 4* and *8-shot* generation tasks. These in-context examples are excerpt from the FOLIO dataset. Thus, the experiments on FOLIO can be considered as *in-distribution* task. Meanwhile, ProofWriter dataset is significantly different and

Model	Setting	Logically Correct (↑)	Logically Incorrect	Syntax Error (↓)	F1 Overall (↑)	F1 True Label (↑)
GPT-3.5 - LINC	1-shot	33.30	15.44	51.26	40.49	22.71
	2-shot	45.82	23.24	30.93	38.67	52.32
	4-shot	51.81	24.45	23.74	45.50	59.10
	8-shot	51.31	24.78	23.90	45.63	53.29
GPT-4 - LINC	8-shot	**64.01**	23.08	12.91	59.89	**67.00**
Llama-3 8B Instruct	2-Shot	50.15	27.99	21.86	55.43	54.11
	SFT	54.26	34.26	11.47	56.73	58.84
	SFT + DPO	50.98	29.51	19.51	55.80	56.47
	SFT + KTO	55.15	30.98	13.87	59.56	61.74
Gemma-2 2B Instruct	SFT	49.71	34.80	15.49	53.12	54.51
	SFT + DPO	34.95	21.23	43.82	44.09	42.30
	SFT + KTO	50.93	29.80	19.26	56.02	54.39
Phi-3.5 Mini Instruct (4B)	SFT	58.43	31.08	10.49	61.47	62.03
	SFT + DPO	61.13	31.08	**7.79**	63.59	60.73
	SFT + KTO	**61.52**	29.41	9.07	**64.43**	**63.76**

Table 3: Results on NL to FOL conversion on the FOLIO validation dataset. The results are an average of 10 runs.

Model	Setting	Logically Correct (↑)	Logically Incorrect	Syntax Error (↓)	F1 Overall (↑)	F1 True Label (↑)
Llama-3 8B Instruct	SFT	46.14	42.39	11.47	48.04	38.80
	SFT + DPO	45.31	35.47	19.22	50.16	45.55
	SFT + KTO	48.58	38.06	13.36	51.58	44.12
Gemma-2 2B Instruct	SFT	46.78	32.56	20.67	51.74	45.91
	SFT + DPO	39.97	22.50	37.53	48.69	42.12
	SFT + KTO	48.86	32.94	18.19	53.65	49.47
Phi-3.5 Mini Instruct (4B)	SFT	61.11	30.0	8.89	63.49	54.22
	SFT + DPO	55.00	36.00	9.00	56.44	46.04
	SFT + KTO	**65.28**	27.89	**6.83**	**67.43**	**58.96**

Table 4: Results on NL to FOL conversion on the ProofWriter dataset. These models are trained on the FOLIO dataset. The ProofWriter dataset is only used for evaluation. The results are an average of 10 runs.

Model	Setting	Logically Correct (↑)	Logically Incorrect	Syntax Error (↓)	F1 Overall (↑)	F1 True Label (↑)
Llama-3 8B Instruct	SFT	27.2	65.0	7.8	41.79	45.98
	SFT + DPO	45.2	48.4	6.4	61.48	62.50
	SFT + KTO	47.4	45.4	7.2	63.18	64.14
Gemma-2 2B Instruct	SFT	49.2	44.6	6.2	64.14	61.77
	SFT + DPO	40.4	29.4	30.2	56.19	55.53
	SFT + KTO	56.4	33.4	10.2	70.23	68.45
Phi-3.5 Mini Instruct (4B)	SFT	87.8	11.6	**0.6**	92.81	92.65
	SFT + DPO	80.2	18.6	1.2	88.32	87.37
	SFT + KTO	**89.0**	10.4	**0.6**	**93.68**	**93.93**

Table 5: Results on NL to FOL conversion on the ProntoQA validation dataset. These models are trained on the FOLIO dataset. The PronotQA dataset is only used for evaluation. The results are an average of 10 runs.

thus requires generalizing *out-of-distribution*.

4.3 Fine-tuned Variants of LLMs

We use the following models for our experiments: Llama-3 (Grattafiori et al., 2024), Gemma-2 (Team et al., 2024), Phi 3.5 (Abdin et al., 2024) We follow a two-stage fine-tuning approach: (i) fine-tuning these models using $\mathcal{D}_{SFT}$ dataset for SFT task and then (ii) fine-tuning them with our $\mathcal{D}_{Pref}$ dataset using preference optimization methods such as DPO (Rafailov et al., 2024) and KTO (Ethayarajh et al., 2024).

4.4 Main Results

FOLIO From Table 3 & 4, we see almost all of our SFT models perform close-to and better than

the best GPT3.5 baselines. Overall, the Phi3.5 Mini SFT version outperforms the other SFT models by rest of the models by atleast **4.17%**, and GPT-3.5 LINC 8-shot by 7% in the logically correct metric. The syntax error of Phi3.5 Mini SFT is also 13% less than the best GPT-3.5 LINC model and 2% less than the GPT-4 LINC model.

Preference optimization with KTO also leads to further improvement in performance. We reach 61.52% logical accuracy and 64.43% overall F1 in FOLIO. In general, we observe SFT + KTO always leads to improvement in performance compared to SFT, which is not the case for SFT + DPO vs SFT.

ProofWriter and ProntoQA We evaluate the models trained on the FOLIO dataset on the ProofWriter and ProntoQA dataset. We see a similar trend in results for these other two evaluation datasets, where SFT + KTO version of Phi3.5 Mini reaches the highest performance. We reach 65.28% and 89.0% logical correctness in ProofWriter and ProntoQA dataset, respectively. Llama and Gemma models show results in similar range for the ProofWriter dataset. However, for ProntoQA, the best version of Gemma significantly outperform the best LLama version. Overall, our dual fine-tuning approach of SFT and preference optimization shows great promise towards accurate conversion of NL problems into FOL.

5 Analysis and Ablation Studies

5.1 Evaluating Semantic Content of Translated FOLs

We follow an autoencoder approach, converting the premise stories to FOLs and converting them back to textual paragraphs for evaluation. We evaluate the FOLs generated by comparing the sentence embeddings of the FOL-generated paragraph and the input premise paragraphs by cosine similarity. Further, the similarities between the generated NLs and premise paragraphs show the information carried while encoding to and decoding from the FOL representations. We follow different levels of evaluation to make space for jumbled NLs since the premise story need not follow a particular order of sentences. Thus, we evaluate the generated NLs against the premise paragraph in three methods: (i) Firstly, we directly compare the paragraph level embeddings of the premise paragraph and the generated NL. (ii) We allow for jumbling and take the average pair-wise similarity of the two. (iii) Lastly, we allow for jumbling while taking the average of maximum sentence similarity between pairs of sentences between the two.

Evaluation metric	LLAMA3	GPT3.5
NL sentence mean similarity	54.83	55.85
NL sentence max similarity	84.09	84.67
NL paragraph similarity	89.13	89.22
FOL sentence mean similarity	54.83	54.57
FOL sentence max similarity	84.09	79.35
FOL paragraph similarity	86.22	87.40

Table 6: Evaluating semantic content of translated FOLs for LLAMA3 and GPT3.5 through an autoencoder approach

Following this approach, we find that both LLAMA3 and GPT3.5 perform similarly with a paragraph average similarity of **89.13%** and **89.26%** respectively. Further results for each of the evaluation methods can be found in table Table 6.

5.2 Qualitative analysis

From the results reported in Tab.s 3 & 4, it is quite evident that our fine-tuned models are significantly better in all evaluation criteria. We further explore *How the generations from fine-tuned models differ from other neurosymbolic models?* We focus on comparing Llama-3 8b instruct+SFT+KTO model vs. Phi-3.5 Mini+SFT+KTO as their performance exceeds other models.

Qualitatively, we find that both the models share few similarities and dissimilarities in their generations and highlight some key findings from both. We also compare the model errors with the different failure modes for Linc reported in (Olausson et al., 2023). Using the same notation L1, L2 and L3 corresponding to implicit information loss, explicit information mistakes and syntax errors correspondingly.

Similarity S_1: Lack of consistent usage of predicates Often, the models use different predicate to convey the same meaning in subsequent sentences leading to loss of information to Prover9. Nevertheless, finetuned LLama-3 suffers less from this type of errors since we see more consistent usage of predicates in LLama. For example, in the snippet below:

Premise 1: Susan flies to LGA airport.
FOL: $FliesTo(Susan, LGAAirport)$
Premise 2: The departure and arrival can not be the same airport.
FOL_Llama: $\neg EqualAirports(Daniel, Susan)$

Model	Setting	Small Context		Medium Context		Large Context	
		Overall Acc. (↑)	True F1	Overall Acc. (↑)	True F1	Overall Acc. (↑)	True F1
Llama-3 8B Instruct	SFT	59.76	62.93	55.07	59.96	19.00	21.05
	SFT + DPO	65.00	68.67	48.95	55.77	21.00	17.54
	SFT + KTO	66.19	71.19	55.00	61.64	25.00	22.22
Gemma-2 2B Instruct	SFT	47.86	50.25	50.59	55.96	44.00	46.43
	SFT + DPO	44.29	42.00	32.17	41.60	38.00	52.46
	SFT + KTO	57.38	57.02	49.61	53.66	44.00	55.07
Phi-3.5 Mini Instruct	SFT	58.10	63.77	59.08	60.90	50.00	72.73
	SFT + DPO	58.33	60.75	61.25	59.48	71.00	77.14
	SFT + KTO	58.33	63.30	62.96	64.74	53.00	50.79

Table 7: Analysis of the performance of various models across different input context sizes in the validation set of FOLIO. We group the instances in small context (1-2 sentences), medium context (3-5 sentences) and large context (more than 5 sentences). The Overall Acc. column corresponds to the logical correctness metric across the full validation set.

FOL_Phi: $\neg(DepartFrom(x) \wedge ArriveAt(x))$
TEXT: John flies from LGA airport.
FOL: $liesFrom(John, LGAAirport)$
TEXT: Susan flies from LGA airport.
FOL: $liesFrom(Susan, LGAAirport)$

Similarity S_2: Both models suffer with complex logic As expected, both the models fare poorly with large sentences with more complex logic. In the example below, both the models suffer in the same way confusing neither-nor logic:

TEXT: If Rock is neither a fly nor a bird, then Rock neither flies nor breathes.
FOL: $\neg((Fly(Rock) \vee Bird(Rock)) \Rightarrow (\neg Fly(Rock) \wedge -Breathes(Rock)))$

Similarity S_3: Problem with Uncertain labels We find the accuracies related to *Uncertain* label highly unreliable due to the high number of ways in which *Uncertain* label can be reached. We have found the following methods the LLM models utilize in order to unexpectedly reach the *Uncertain* label. The various ways are as follows:

- The LLM has an inconsistent usage of predicates, like S_1.
- Incorrect translation of a single FOL in the story causing loss of information.

Analysis of NL to FOL Conversion across Input Lengths We analyze how well the fine-tuned models convert NL to FOL stories across different input context sizes in Section 5.1. We group the FOLIO dataset into three categories – instances with small (1-2 sentences), medium (3-5 sentences) and large context (more than 5 sentences). Llama and Gemma models shows monotonically decreasing performance as we increase the context length. Interestingly, the Phi models do almost similarly in the small and medium context instances, which is not the case for the other two models.

Model	L1	L2	L3	Wrong Translation
Llama-3 8B Instruct	10	15	14	9
Phi-3.5 Mini Instruct	13	20	14	1

Table 8: Number of instances in FOLIO, with errors corresponding to errors from LINC. L1: Implicit Information Loss, L2: Explicit information errors, L3: Syntax Errors

Error modes of LINC According to (Olausson et al., 2023), there are mainly 3 modes of failure for neurosymbolic solvers like LINC. Thus, we map the failure methods for our best models LLAMA-3 and PHI3 to investigate whether preference optimization helps alleviate these issues. We note our results in Table 8. Thus, we can deduce that models still suffer while representing explicitly mentioned information due to the choices of predicates, paving way for future improvements.

6 Conclusion

In this work, we present an efficient method for improving logical reasoning of LLMs through preference optimization on a synthetically generated dataset. Our experiments show that preference optimization on this dataset leads to significant performance gains in all of our evaluation criteria. Furthermore, carrying out a qualitative and quantitative analysis of our models shows the various advantages and shortcomings of our approach. This work thus shows promise in the field of LLMs as parsers through preference optimization. Paving the way for future work on continual fine-tuning of neurosymbolic solvers for logical reasoning.

Limitations

Our work is among the first ones which attempts to convert natural language reasoning problems holistically to an equivalent logical representation in

First Order Logic. The primary limitations of the work is as follows.

1) At various stages of dataset creation, we depend on the predicted logical label. However, it is not guaranteed that if the logical label is correct, the program will also be correct. While we attempt to evaluate the semantic content, this is clearly an open problem and requires further exploration.

2) We only explore English-FOL as a representative natural-formal language pair combinations. Provided the current failure modes of LLMs, it is probable that parsing errors will be higher as we change to even low-resource formal languages or low-resource natural language. Many low-resource formal languages have been shown to be useful such as LEAN for mathematical theorem proving (was used for GPT-4's math olympiad work). One can possibly adopt our framework for generalizing to such languages as well.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and et al. Alon Benhaim. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1210–1219, Baltimore, Maryland. Association for Computational Linguistics.

David Chanin. 2023. Open-source frame semantic parsing. *arXiv preprint arXiv:2303.12788*.

Zhenyun Deng, Yonghua Zhu, Yang Chen, Michael Witbrock, and Patricia Riddle. 2022. Interpretable amr-based question decomposition for multi-hop question answering. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4093–4099. ijcai.org.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning, CoNLL 2005, Ann Arbor, Michigan, USA, June 29-30, 2005*, pages 9–16. ACL.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, and et al. Angela Fan. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. *ArXiv*, abs/2209.04994.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to SQL: where are we today? *Proc. VLDB Endow.*, 13(10):1737–1750.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. 2024. A survey of NL2SQL with large language models: Where are we, and where are we going? *CoRR*, abs/2408.05109.

Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. 2024. Are emergent abilities in large language models just in-context learning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 5098–5139. Association for Computational Linguistics.

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.

W. McCune. 2005–2010. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2024. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *Preprint*, arXiv:2410.21272.

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.

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

Haritz Puerto, Martin Tutek, Somak Aditya, Xiaodan Zhu, and Iryna Gurevych. 2024. Code prompting elicits conditional reasoning abilities in text+code llms. *Preprint*, arXiv:2401.10065.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

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*.

Arpit Sharma, Nguyen Ha Vo, Somak Aditya, and Chitta Baral. 2015. Towards addressing the winograd schema challenge - building and using a semantic parser and a knowledge hunting module. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1319–1325. AAAI Press.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *Preprint*, arXiv:2012.13048.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, and et al. Pouya Tafti. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Ramya Keerthy Thatikonda, Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. 2024. Strategies for improving nl-to-fol translation with llms: Data generation, incremental fine-tuning, and verification. *arXiv preprint arXiv:2409.16461*.

Qi Wu, Peng Wang, Chunhua Shen, Anthony R. Dick, and Anton van den Hengel. 2016. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4622–4630. IEEE Computer Society.

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

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

A Appendix

A.1 Predicate generation through code prompting

Evaluation metric	LLama3	GPT3.5
Direct comparison	0.3925	0.4355
Similarity Pairing	0.7800	0.7766

Table 9

Inspired from direct code generation abilities of text+code LLMs (Puerto et al., 2024), we aim to elicit better reasoning abilities of these LLMs by generating Python-like boolean functions. These Python-like functions are equivalents of predicates in FOL. Using the same predicates provided in **premises-FOL**, we prompt the model to generate Python-like boolean functions. Further, we add comments to provide models with context that is proven to generate better Python code. We evaluate the predicates generated by comparing these with the predicates present in **premises-FOL** in the following ways: (i) Direct comparison: We take

the set-wise intersection of the two. (ii) Similarity Pairing: Since each of the predicate need not be present in the generated code, we find pairwise correlations of the generated predicates with the gold predicates. We consider the two predicates to be similar if one is a substring of another. These results of this experiment are presentine