PROOFFOL: Improving NL-to-FOL Translation of LMs

Anonymous ACL submission

Abstract

While formal language is well-suited for deductive logical reasoning with language models (LM), translating natural language (NL) into first-order logic (FOL) often results in errors that are not explicitly addressed by the literature. In the absence of larger scale NL-FOL translation data, we introduce PROOFFOL, a high-quality FOL-annotated dataset containing $10.4k$ examples, with multiple premises and a conclusion. Additionally, we categorize the FOL translation errors made by LMs and highlight a significant reduction in translation error from models trained on PROOFFOL. The improvement in FOL translation also boosts performance in downstream reasoning by the LMs on several logical reasoning benchmarks.[1]

1 Introduction

The tool-augmented paradigm in logical reasoning tasks with language models (LMs) is to translate statements from natural language (NL) to predicate first-order logic (FOL). These translations are then sent for execution to external solvers such as Z3 (De Moura and Bjørner, 2008) and Prover9 (McCune, 2005), where the tool returns 'True', 'False', 'Uncertain/Unknown' as final output (Ye et al., 2023; Pan et al., 2023; Olausson et al., 2023). Beyond logical reasoning, Autoformalisation methods (Yang et al., 2024a; Wu et al., 2022) have a similar process of conversion.

Recent work by Yang et al. (2024b) has highlighted the systematic errors that even the most advanced LLMs make during translation of a *single* NL statement into its corresponding first-order logic. Realistic logical reasoning scenarios are significantly more demanding, as they involve *multiple* premise statements followed by a conclusion to be verified. These scenarios require consistent NL-to-FOL translations (e.g., in predicate naming or the translation of logical operators) across multiple statements.

Existing approaches to reducing NL-to-FOL translation errors rely on the LLM's ability to understand and self-correct the translation inaccuracies based solely on the error message from an external SMT solver (Pan et al., 2023). However, the ability to comprehend such error messages is often restricted to larger-scale models (e.g., 175B), and is generally unavailable in smaller LMs.

To bridge this gap, we construct PROOFFOL dataset by leveraging the ProofWriter (Tafjord et al., 2021) dataset. Using a format standardised to the Prover9, we first utilise GPT-4o to generate FOL translations. The generated FOLs are passed through a pipeline consisting of format correction, Prover9-based syntax validation, and semantic filtering to retain high-quality translations. This process yields $10{,}424$ examples, each containing multiple premises and a conclusion. Table 1, provides a comparison of PROOFFOL and existing datasets.

Experiments with three language models demonstrate that variations of supervised fine-tuning using PROOFFOL significantly reduce NL-to-FOL translation errors and improve downstream logical reasoning performance across four benchmarks.

2 NL-to-FOL Translation

First-order logic (FOL) is a logical framework using predicates, quantifiers, variables, and logical connectives, often applied to natural language reasoning. Large Language Models (LLMs) have demonstrated varying success in translating natural language into formal representations (see Appendix A for an overview). Among these formalism, NL-to-FOL translation presents unique challenges involving syntactic and semantic interpretations. Here we only attempt to categorize the

[1] Code, data, and few-shot details are available at https://anonymous.4open.science/r/ProofFOL-CD7A/. Models will be released with publication.

Dataset	Type	Overall Statistics			FOL Statistics								
		Multiple Premise	Training Examples	#NL-FOL pairs	Avg. #literals	$\forall$	$\exists$	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\oplus$
LogicNLI	Reasoning	Yes	12000	18960	2.8	2783	5327	10230	6590	2373	8712	3288	0
FOLIO	Reasoning	Yes	1001	2679	2.1	1111	182	421	631	167	1137	17	121
MALLS	Translation	No	27300	27300	4.6	27256	1738	3443	25209	5164	25212	1835	1777
PROOFFOL	Reasoning	Yes	10424	37141	7.1	10232	4	7325	10166	4	10259	1	0

Table 1: Statistics of LogicNLI, FOLIO, MALLS, and PROOFFOL.

syntax errors which we will refer to in our analysis (§4.2).[2] Syntactic errors arise from deviations in grammatical rules during translation. For example, the rule *"every free variable assigned to a predicate must have a quantifier"* applied to the statement "Green people are blue" requires a quantifier '$\forall$' for the variable x. Missing quantifiers or operators can lead to parsing errors, often detected by Prover9, which provide feedback on syntax issues. The most common categories of syntactic errors include:

- *Parsing* errors, caused by missing or invalid operators, incomplete FOL expressions, or incorrect placement of parentheses,
- *NoneType* errors, arising from misplaced quantifiers or sentence-level inconsistencies,
- *Token* errors, involving the use of invalid symbols, and
- *Arities* errors, occurring when a predicate is used with an inconsistent number of arguments across different FOL expressions.

Details and examples are provided in Appendix C.

3 PROOFFOL

3.1 FOL Data Generation

The alignment of a language model to follow instructions for a specific task can be accomplished by fine-tuning on substantial data. The task of formal translations require first-order logic of their natural language counterparts. Ideally, this task is at a passage level rather than sentence level, which makes it challenging for a language model to follow a required format. To overcome this, we need sufficient passage level translations, which are time-consuming to generate through human annotations. We introduce a streamlined process for generating this FOL data and ensuring correctness of the format, grammar, and order of the translations.

PROOFFOL Construction. For the data generation process, we build on ProofWriter which comes with large number of training records, each consisting of multiple premises and a conclusion, and variations in depth of reasoning. We sampled $15k$ records from depth-5 of ProofWriter. The records are sampled randomly ensuring an equal distribution of the labels ($5k$ each); True, False, and Unknown. The format of the FOLs is set to be consistent with the Prover9 theorem prover.[3] The fixed format of demonstrations is adapted from Pan et al. (2023), where we generate predicates followed by first-order logic statements for each sentence. We standardize the formal language to Prover9 and apply it consistently across all translations. Syntax validation is performed using Prover9, which provides distinct feedback for each type of error. During error analysis, we identified grammar limitations that could be addressed by extending the tool to support Unicode decoding, tolerate unordered quantifiers, and allow negation of entire formulas. Incorporating these grammar rules substantially reduced penalization for translations. To identify semantic errors, we compared the ground-truth labels from ProofWriter with the outputs generated by Prover9. Through this process, we obtained $10{,}424$ passage-level records containing FOL statements. See Appendix B for more details on the data generation pipeline.

Quality Evaluation by Human. We conducted a human evaluation on a random sample of 50 records from our dataset (containing a total of $1,018$ individual statements) where the final outputs were correct. Out of these $1,018$ translations, we identified only 22 errors (i.e., 2%), half of which were minor and did not alter the meaning of the

[2] Semantic errors such as incorrect quantifiers, operators, or predicates conform to formal syntax but misrepresent the intended meaning. Unlike syntactic errors, they are not automatically detectable through standard parsers or theorem provers and require mechanisms capable of assessing the semantic equivalence of statements. In the presence of a task such as deductive logical reasoning, the proxy for detecting semantic errors is the final reasoning accuracy (i.e., when there is no syntax error, but the final reasoning outcome from the tool does not match the ground-truth).

[3] We ran a small scale experiment through GPT-4o for various few-shot combinations. We selected the most optimal demonstrations with the least format and translation issues.

FOL statements. We also extracted predicates and applied an arity check along with similarity analysis on lemmatized predicates across all records, finding no errors in predicate values.

Comparison with Existing Datasets Compared to our dataset, PROOFFOL, MALLS (Yang et al., 2024b) provides a similarly large number of NL-FOL translations but only targets single-statement generations, limiting its applicability in symbolic reasoning for logic. Previous deductive logic datasets, such as FOLIO (Han et al., 2024) and LogicNLI (Tian et al., 2021), provide NL-FOL pairs for logical reasoning and natural language inference tasks, respectively. However, these datasets are constrained by their training data size, the total number of NL-FOL pairs, and the average number of literals (Table 1). Given the large number of premise–conclusion pairs with their corresponding first-order logic translations, PROOFFOL proves to be one of the largest datasets currently available.

3.2 Supervised Fine-Tuning and Inference

SFT. We employ Supervised Fine-Tuning (SFT) to generate predicates, followed by the first-order logic representations (FOL) of the premises and conclusion. Each input x to the SFT model consists of a list of n premise statements $P_x = [P_1, P_2, \ldots, P_n]$ and a conclusion C_x. The model outputs the corresponding predicates and FOLs of the premises and conclusion, $[Pred_x, FOL_{P_x}, FOL_{C_x}]$.

Decompositional SFT. While the standard SFT approach involves feeding the entire input and training the model to generate the full output, we also adopt a decompositional formulation by breaking the task into simpler steps. Given x, we first predict all predicates $Pred_x$. Then every natural language statement NL_i of x (whether a premise or conclusion) is paired with $Pred_x$ as $[NL_i, Pred_x]$, and the output is its corresponding $[FOL_i]$.

Inference. The LM is provided with the NL premises and conclusion to generate the predicates and FOL translations for the complete input.

4 Experiments

We fine-tuned `Qwen2.5-3B-Instruct` (Hui et al., 2024), `Gemma-3-4B-Instruct` (Kamath et al., 2025), and `Phi-4-mini-Instruct` (Abouelenin et al., 2025) on FOL translations from ProofFOL and FOLIO. We conducted fine-tuning on individual datasets and their combination with and witout decomposition (denoted as SFT_{dec} and SFT_{+} respectively). All models were trained using 3 epochs of supervised fine-tuning with LoRA (Hu et al., 2022). Due to the small size of FOLIO (1,001 training records), 3 epochs were insufficient to achieve meaningful performance; therefore, we extended it to 10 epochs (denoted as SFT_{fo}). For inference, we used 8-bit quantization with a temperature of 0.1 and a maximum token limit of 1,000.

		Qwen-2.5-3B		Gemma-3-4B		Phi-4-mini	
	Type	ExcRate	Acc	ExcRate	Acc	ExcRate	Acc
ProofWriter	ICL	57.83	39.67	66.33	59.68	74.50	57.67
	SFT	**99.83**	**97.00**	88.66	85.83	90.00	87.67
	SFT_{fo}	31.33	22.33	70.33	51.00	51.66	38.67
	SFT_{+}	96.83	93.33	98.83	96.83	**100**	**97.83**
	SFT_{dec}	56.83	54.50	**99.50**	**97.50**	94.17	91.67
ProntoQA	ICL	6.80	0.60	13.60	1.80	6.00	2.60
	SFT	26.4	2.40	6.40	0.60	10.80	2.40
	SFT_{fo}	**63.40**	18.60	62.40	38.00	53.20	15.60
	SFT_{+}	55.60	**21.60**	51.80	23.20	44.60	**30.60**
	SFT_{dec}	25.40	13.80	**89.60**	**53.60**	**63.00**	28.60
ProverQA	ICL	0.20	0.20	30.80	9.40	47.20	18.40
	SFT	11.00	3.80	1.00	0.20	19.20	7.20
	SFT_{fo}	21.60	8.80	41.00	14.40	25.20	10.40
	SFT_{+}	**50.40**	**22.20**	**65.80**	**33.60**	42.20	24.40
	SFT_{dec}	25.20	10.00	41.20	19.80	**59.60**	**28.00**
FOLIO	ICL	22.16	12.81	46.79	27.09	42.85	30.05
	SFT	20.68	8.37	13.79	2.46	8.86	5.42
	SFT_{fo}	**57.63**	**35.47**	64.04	34.98	49.26	29.06
	SFT_{+}	57.63	30.05	**66.00**	**36.95**	**61.08**	**38.92**
	SFT_{dec}	12.81	6.90	64.53	40.39	59.60	37.44

Table 2: Comparison of models' deductive reasoning accuracy under FOL-based output prediction and the execution rate (%) of the FOLs by the tool. ICL denotes in-context learning (3-shots); SFT denotes supervised fine-tuning (on PROOFFOL); SFT_{fo} denotes SFT on FOLIO; SFT_{+} and SFT_{dec} denote SFT and decompositional SFT on PROOFFOL and FOLIO combined, respectively.

Benchmarks. We report the downstream accuracy of the fine-tuned models on four deductive logical reasoning benchmarks: ProofWriter (Tafjord et al., 2021), ProntoQA (Saparov and He, 2023), ProverQA (Qi et al., 2025a), and FOLIO (Han et al., 2024). Notably, ProverQA does not include FOL translations in its training data, and ProntoQA is a test-only benchmark, making inference on these two datasets out-of-distribution for our models. For ProofWriter and ProntoQA, we use the test set provided by Pan et al. (2023), consisting of 600 and 500 records, respectively. For FOLIO, we used 203 development records for test. For ProverQA, we used 500 records of the 'hard' subset of the test set.

Evaluation. We evaluate models in two modes:

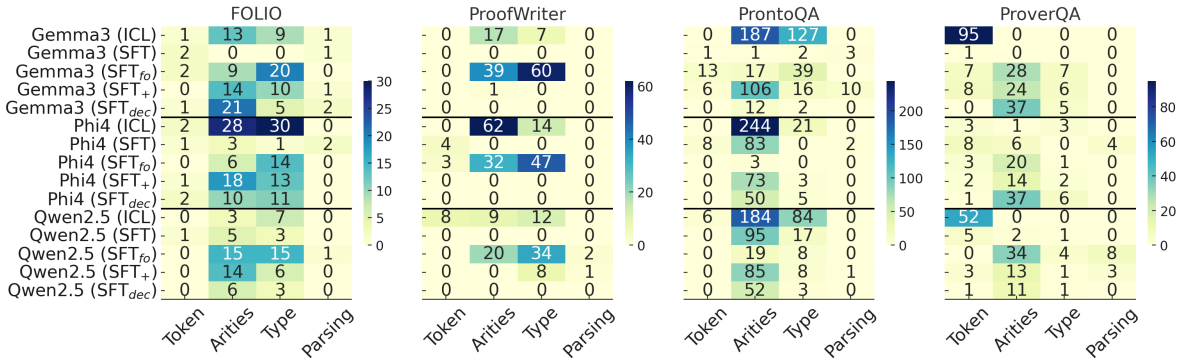


Figure 1: Heatmap of error types counts across datasets, models, and methods.

In-Context Learning (ICL), using three randomly selected few-shot examples from the training sets, and SFT for NL-to-FOL translation. In both modes, the models generate FOL translations of the natural language inputs, and the final logical outcomes are derived using Prover9. For ProntoQA, which is a test-only dataset, we use demonstrations from ProofWriter for ICL inference, while all other datasets use in-domain few-shots.

4.1 Main Results

Table 2 presents the results on logical reasoning datasets using both ICL and SFT, comparing performance across different model families with smaller parameter sizes on various benchmarks. We also report the execution rate—the percentage of valid records—to track syntactic errors in the generated FOL translations. As the results indicate substantial variation across models, we analyze their performance separately for all benchmarks. **`Qwen2.5-3B-Instruct`** shows relatively low performance with ICL across all benchmarks. Fine-tuning on in-domain datasets improves results: performance on ProofWriter increases when trained on ProofFOL, and performance on FOLIO improves when trained on FOLIO. Notably, Qwen shows additional performance gains on out-of-distribution benchmarks when fine-tuned on the combined dataset, suggesting that mixed-domain training enhances generalization. **`Gemma-3-4B-Instruct`** and **`Phi-4-mini-Instruct`** demonstrate stronger ICL performance compared to Qwen. Both models achieve consistent improvements in execution rate and accuracy when fine-tuned on the combined dataset (ProofFOL + FOLIO) compared to individual datasets, highlighting the benefit of mixed-data training for SFT. The decompositional method leads to a notable performance boost for the Gemma models, while Qwen models appear less suitable for this approach. ProntoQA generally benefits from decompositional-based fine-tuning across most cases. Finally, although the execution rate for ProntoQA remains high for SFT$_{fo}$ across all models, the variability in accuracy indicates that these FOLIO-trained models are still prone to semantic errors on this benchmark.

4.2 Error Distribution

We present a categorized distribution of errors across all benchmarks and highlight several notable patterns (see Figure 1). Formatting errors capture the model's ability to produce outputs in the correct format, reflecting format-following capability, though this aspect is beyond the scope of our current discussion. The smaller LMs exhibit a high frequency of Arity and Type errors under ICL. Fine-tuning on ProofFOL alone leads to substantial reductions in these errors for FOLIO, ProofWriter, and ProverQA. Although models fine-tuned on FOLIO alone achieve modest accuracy gains, our error analysis reveals that leveraging our dataset—either independently or in combination—significantly reduces syntactic errors. Compared to SFT$_{+}$, SFT$_{dec}$ shows a reduction in syntax errors; however, it also results in a higher incidence of formatting errors. Detailed results and further discussion are provided in Appendix D.

5 Conclusion

In this paper, we provide a systematic analysis of common translation errors made by LMs when used for natural language to first-order logic (NL-to-FOL) translation. Using PROOFFOL, we fine-tuned a set of smaller LMs, achieving improved performance over in-context learning. Furthermore, our error analysis demonstrates the impact of this dataset in reducing errors across most benchmarks, both in-domain and out-of-domain.

Limitations

PROOFFOL is generated on top of ProofWriter which will inherit its limitations in terms of diversity and complexity of the statements. These limitations reveal themselves in the lower diversity of the generated FOLs for logical operators other than $\forall, \neg, \wedge, \rightarrow$ (Table 1). Additionally, the silver data generated using `GPT-4o`, though processed with care, may contain minor inconsistencies as pointed in §3.1.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Michael K Chen, Xikun Zhang, and Dacheng Tao. 2025. Justlogic: A comprehensive benchmark for evaluating deductive reasoning in large language models. *arXiv preprint arXiv:2501.14851*.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej 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. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucinska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, and Ivan Nardini. 2025. Gemma 3 technical report. *CoRR*, abs/2503.19786.

Mingyue Liu, Ryo Ueda, Zhen Wan, Katsumi Inoue, and Chris G. Willcocks. 2024. Neuro-symbolic contrastive learning for cross-domain inference. In *Proceedings 40th International Conference on Logic Programming, ICLP 2024*, volume 416 of *EPTCS*, pages 78–94.

Long Hei Matthew Lam, Ramya Keerthy Thatikonda, and Ehsan Shareghi. 2024. A closer look at tool-based logical reasoning with LLMs: The choice of tool matters. In *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association*, pages 41–63, Canberra, Australia. Association for Computational Linguistics.

William McCune. 2005. Release of Prover9. In *Mile high conference on quasigroups, loops and nonassociative systems, Denver, Colorado*.

Maxwell I. Nye, Michael Henry Tessler, Joshua B. Tenenbaum, and Brenden M. Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 25192–25204.

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 Conference on Empirical Methods in Natural Language Processing, EMNLP, pages 5153–5176.

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

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. 2025a. Large language models meet symbolic provers for logical reasoning evaluation. In *The Thirteenth International Conference on Learning Representations*.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. 2025b. Large language models meet symbolic provers for logical reasoning evaluation. *arXiv preprint arXiv:2502.06563*.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics.

Paul Tarau. 2024. On llm-generated logic programs and their inference execution methods. In *Proceedings 40th International Conference on Logic Programming, ICLP*, volume 416 of *EPTCS*, pages 1–14.

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.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 32353–32368.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. 2024a. Formal mathematical reasoning: A new frontier in ai. In *arXiv preprint arXiv:2412.16075*.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024b. 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. 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, NeurIPS*.

A Related Work

LMs for Symbolic Translation. The use of formal language translations by LLMs was initially attempted by Nye et al. (2021), with an intent to emphasize the importance of dual process theory for logical reasoning tasks. Following this, the process of reasoning was offloaded to theorem provers and LLMs served as systems to generate logic programs for deductive reasoning (Matthew Lam et al., 2024; Pan et al., 2023; Ye et al., 2023; Olausson et al., 2023) and inference tasks (Liu et al., 2024; Tarau, 2024). The available research in this method majorly differs in variation of formal language (used by different theorem provers) and verification process to handle translation errors. These methods make use of expensive and ambiguous GPT model (we refer to ambiguity in the data used to train the GPT model), restricting the symbolic framework to non-critical domains. Corresponding to the work in formal logic, Yang et al. (2024b) applied supervised finetuning to LLaMA model to improve the natural language to first-order logic translations at a sentence level. Our research shifts the focus to building a complete translation system that can handle multiple statements. In addition to this, we perform a systematic analysis of translation errors.

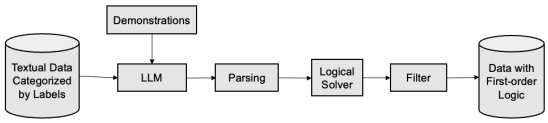
Deductive Reasoning Benchmarks. Deductive logical reasoning requires logical derivation of conclusion using a set of premises. Datasets such as ProofWriter (Tafjord et al., 2021), ProntoQA (Saparov and He, 2023), ProverQA (Qi et al., 2025b), and JustLogic (Chen et al., 2025) highlight multi-hop reasoning paths, but limit their use in tasks requiring formal rigour [4]. To address this, we build an FOL dataset to adapt ProofWriter for formal reasoning tasks, enhancing its ability to handle structured, interpretable logical inferences and enabling access to verification and theorem proving.

FOLIO (Han et al., 2024), with human-annotated FOL sentences, offers a semantically complex deductive task but is limited by its small size, limiting its use for developing or improving formal language translation models. We present data augmentation technique to overcome this issue.

[4] Although these datasets are generated using logic programs, the original logical forms of the data and framework used for conversion to natural language are not released with their data and therefore not available.

Label	Count
True	3587
False	3441
Unknown	3396

Table 3: Label Distribution in PROOFFOL



Data	Total Train	Syntax Errors	Output Mismatch	Final Train
ProofWriter	15,000	3881	695	10,424

Figure 2: Overview of the Data Generation Pipeline and Key Statistics.

B More Details on PROOFFOL

The data pipeline applied for generating ProofFOL is detailed in Fig 2. The train data is 15,000 data points with depth-5 from ProofWriter dataset. The 15k records are sampled randomly ensuring a fair distribution of the labels; True, False, and Uncertain. The LLM here is GPT4o with a output token length of 1,000 for each generation. We use $url = /v1/chat/completions$ format for batch generations of GPT4o (see Figure 3 for prompt details). The logical solver is Prover 9, a theorem prover suitable to run in python environment with nltk library, that uses CNF conversions, quantifier operations, and skolemization to transform the clauses into a tree format. Parsing errors by Prover 9 occur when the FOL formula cannot be converted to a tree structure because it does not follow specific grammar rules. After filter and parsing stages, we get 10,424 records with FOL statements with a fair distribution of labels (Table 3). Syntax errors are the errors thrown by the tool and semantic errors are measured by comparing the ground truth label with the solver output.

C Syntax Error Analysis

Table 4 shows examples of each type of syntax error. These are sub-categorizes by manually analysing the FOL statements where there was feedback from the tool. The details of each error are provided here:

- Missing quantifier: When a predicate includes a variable, it must have either a Universal Quantifier '∀' or an Existential Quantifier '∃'. This error occurs if either quantifier is missing.

	Type	Example
Parsing	Missing Quantifier	BerkeleyCollege(x) $\wedge$ResidentialCollegeAt(x, yaleUniversity)
	Parenthesis error	BeneficialTo(cherry, people) $\oplus$On(cherry, warningList)) $\rightarrow \neg$RedFruit(cherry)
	Completion error	$\forall x$(Athlete(x) $\rightarrow\neg$NeverExercises(x)) Never: does not exist a time
Type	Quantifier location	$\exists y$(Own(emily, y) $\wedge$Roommate(y)) $\rightarrow$ $\exists y$(Own(emily, y) $\wedge$LiveIn(emily, apartment))
	Missing variable	$\forall x \exists y$(In(indonesia) $\wedge$ Prosecutor(x) $\wedge$ SpecialCrime(y) $\rightarrow$ InvestigatePersonally(x, y))
Token	Special token	Endowment(yale, 42.3 billion)
	Unknown Operator	$\forall x$(Rating(x, y) $\wedge$ $y \geq 4 \rightarrow$ Listed(x))
Arities	Arities errors	Error: Sees(Tiger, Mouse); $\forall x$(((Visits(x, Rabbit)) $\wedge$ (Sees(Mouse))) $\rightarrow$ (Visits(x, Tiger))) True: Sees(Tiger, Mouse); $\forall x$ (((Visits(x, Rabbit)) $\wedge$ (Sees(x, Mouse))) $\rightarrow$ (Visits(x, Tiger)))
	Subject Predicate	Platypus(platypus) $\wedge\neg$Teeth(platypus) $\wedge$Mammal(platypus)

Table 4: Syntax Errors in First-Order Logic Translations. This table categorizes errors by their cause, with the *underlined* text highlighting the specific cause or location of each error.

Error Category	Representative Pattern
NoneType Error	'NONETYPE' OBJECT HAS NO ATTRIBUTE 'RSTRIP'
Arity Error	THE FOLLOWING SYMBOLS ARE USED WITH MULTIPLE ARITIES
Parsing Error	A TERM CANNOT BE CONSTRUCTED FROM THE MARKED STRING
Token Error	UNEXPECTED TOKEN:'-'

Table 5: Common error categories and representative patterns used to extract LM-generated errors

- Parenthesis error: The formula becomes invalid if there is an extra or a missing parenthesis.
- Completion error: The FOL is either incomplete or contains additional text that disrupts its logical flow.
- Quantifier location: Quantifiers that are either repetitive or incorrectly positioned result in grammatical inaccuracies in the expression.
- Missing variable: When multiple quantifiers are present, the tool fails to parse predicates that lack free variables.
- Special token: The tool does not handle special characters in the input.
- Unknown operator: The tool does not support parsing mathematical equations.
- Arities error: It occurs when predicates are inconsistently applied with a varying number of constants across different statements. Such discrepancies can introduce ambiguity in logical inferences. This is a semantic error that is captured by the tool.
- Subject predicate: The logic is flawed when a subject is used both as a predicate and a constant in the expression.

Error Categorization. The error categories are derived from the feedback provided by the tool. We categorize these errors based on common patterns observed in language model generations (Table 5). Formatting errors are grouped together as cases where the FOL statements never reach the tool, as these generations have incompatible or invalid FOL formatting.

D Error Distribution in FOL Generations

Table 6 reports the statistics of syntax errors. The majority of errors are predicate arity and type errors. Fine-tuning on ProofFOL substantially reduces these errors on in-domain dataset, while combining FOLIO with ProofFOL yields further reductions across all benchamrks. Notably, the **Qwen2.5-3B-Instruct** model shows consistent improvements

Dataset	Model	Config	Token	Arities	Type	Parsing	Formatting
FOLIO	Gemma3 4B Instruct	ICL	1	13	9	1	84
FOLIO	Gemma3 4B Instruct	ProofFOL	2	↓0	↓0	1	172
FOLIO	Gemma3 4B Instruct	FOLIO	2	9	20	0	42
FOLIO	Gemma3 4B Instruct	Mixed	0	14	10	1	44
FOLIO	Gemma3 4B Instruct	Decompositional	1	21	5	2	43
FOLIO	Phi4 mini Instruct	ICL	2	28	30	0	56
FOLIO	Phi4 mini Instruct	ProofFOL	1	↓3	↓1	2	178
FOLIO	Phi4 mini Instruct	FOLIO	0	6	14	0	83
FOLIO	Phi4 mini Instruct	Mixed	1	18	13	0	47
FOLIO	Phi4 mini Instruct	Decompositional	2	10	11	0	59
FOLIO	Qwen2.5 3B Instruct	ICL	0	3	7	0	148
FOLIO	Qwen2.5 3B Instruct	ProofFOL	1	↑5	↓3	0	152
FOLIO	Qwen2.5 3B Instruct	FOLIO	0	15	15	1	55
FOLIO	Qwen2.5 3B Instruct	Mixed	0	14	6	0	66
FOLIO	Qwen2.5 3B Instruct	Decompositional	0	6	3	0	168
ProofWriter	Gemma3 4B Instruct	ICL	0	17	7	0	178
ProofWriter	Gemma3 4B Instruct	ProofFOL	0	↓0	↓0	0	68
ProofWriter	Gemma3 4B Instruct	FOLIO	0	39	60	0	79
ProofWriter	Gemma3 4B Instruct	Mixed	0	1	0	0	6
ProofWriter	Gemma3 4B Instruct	Decompositional	0	0	0	0	3
ProofWriter	Phi4 mini Instruct	ICL	0	62	14	0	77
ProofWriter	Phi4 mini Instruct	ProofFOL	4	↓0	↓0	0	56
ProofWriter	Phi4 mini Instruct	FOLIO	3	32	47	0	208
ProofWriter	Phi4 mini Instruct	Mixed	0	0	0	0	0
ProofWriter	Phi4 mini Instruct	Decompositional	0	0	0	0	35
ProofWriter	Qwen2.5 3B Instruct	ICL	8	9	12	0	224
ProofWriter	Qwen2.5 3B Instruct	ProofFOL	0	↓0	↓0	0	1
ProofWriter	Qwen2.5 3B Instruct	FOLIO	0	20	34	2	356
ProofWriter	Qwen2.5 3B Instruct	Mixed	0	0	0	0	19
ProofWriter	Qwen2.5 3B Instruct	Decompositional	0	0	0	0	259
ProntoQA	Gemma3 4B Instruct	ICL	0	187	127	0	118
ProntoQA	Gemma3 4B Instruct	ProofFOL	1	↓1	↓2	3	461
ProntoQA	Gemma3 4B Instruct	FOLIO	13	17	39	0	119
ProntoQA	Gemma3 4B Instruct	Mixed	6	106	16	10	103
ProntoQA	Gemma3 4B Instruct	Decompositional	0	12	2	0	38
ProntoQA	Phi4 mini Instruct	ICL	0	244	21	0	205
ProntoQA	Phi4 mini Instruct	ProofFOL	8	↓83	↓0	2	353
ProntoQA	Phi4 mini Instruct	FOLIO	0	3	0	0	231
ProntoQA	Phi4 mini Instruct	Mixed	0	73	3	0	201
ProntoQA	Phi4 mini Instruct	Decompositional	0	50	5	0	130
ProntoQA	Qwen2.5 3B Instruct	ICL	6	184	84	0	192
ProntoQA	Qwen2.5 3B Instruct	ProofFOL	0	↓95	↓17	0	256
ProntoQA	Qwen2.5 3B Instruct	FOLIO	0	19	8	0	156
ProntoQA	Qwen2.5 3B Instruct	Mixed	0	85	8	1	128
ProntoQA	Qwen2.5 3B Instruct	Decompositional	0	52	3	0	318
ProverQA	Gemma3 4B Instruct	ICL	95	0	0	0	251
ProverQA	Gemma3 4B Instruct	ProofFOL	1	↓0	↓0	0	494
ProverQA	Gemma3 4B Instruct	FOLIO	7	28	7	0	253
ProverQA	Gemma3 4B Instruct	Mixed	4	24	6	0	137
ProverQA	Gemma3 4B Instruct	Decompositional	0	37	5	0	252
ProverQA	Phi4 mini Instruct	ICL	3	1	3	0	257
ProverQA	Phi4 mini Instruct	ProofFOL	8	↑6	↓0	4	386
ProverQA	Phi4 mini Instruct	FOLIO	3	20	1	0	350
ProverQA	Phi4 mini Instruct	Mixed	2	14	2	0	271
ProverQA	Phi4 mini Instruct	Decompositional	1	37	6	0	158
ProverQA	Qwen2.5 3B Instruct	ICL	52	0	0	0	447
ProverQA	Qwen2.5 3B Instruct	ProofFOL	5	↑2	↑1	0	437
ProverQA	Qwen2.5 3B Instruct	FOLIO	0	34	4	8	346
ProverQA	Qwen2.5 3B Instruct	Mixed	3	13	1	3	228
ProverQA	Qwen2.5 3B Instruct	Decompositional	1	11	1	0	361

Table 6: Error type counts across datasets, models, and methods.

across all categories when trained on ProofFOL, providing strong evidence for the quality of our dataset.

E Few Shot Examples

The Fig 3 shows the format of the few-shot example for generation of FOLs in PROOFOL. We tried this with 2 examples for 50 unique training data points, but it did not generate better translation results. Instead, we ensured that all operators are covered in this example, specifically negation. We first generate predicates associated with the sentence. This is followed by FOL generation.

The prompt for the ICL FOL generation from LMs is presented in Figure 4. The FOL syntax is fixed to represent Prover9 format. We use 3-shots as demonstrations in all ICL experiments.

F Further Details of SFT and Inference

The models are fine-tuned for three epochs with a fixed batch size of 64, a learning rate of 1×10^{-4}, a weight decay of 0.01, and a LoRA configuration (Hu et al., 2022) adapted to the language model on a single A100 GPU. Training uses a causal language modeling loss applied to inputs formatted as `{NL \n\n Predicates; FOL}`, with gradients computed over the entire input.

Inference is performed using 8-bit quantization with the trained LoRA adapter, a temperature of 0.1, and a variable `maximum_new_tokens` value depending on the dataset. PROOFWRITER and PROVERQA require a larger number of tokens during inference compared to FOLIO and PRONTOQA due to their longer input sizes.

G Further details on Language Models

We use three instruction-tuned language models from different model families. All model weights are downloaded from Hugging Face[5]: `Qwen/Qwen2.5-3B-Instruct`, `google/gemma-3-4b-it`, and `microsoft/Phi-4-mini-instruct`. The QWEN model has 3B parameters, while GEMMA and PHI each have 4B parameters.

[5] https://huggingface.co

Few-shot Example with Instruction for Data Generation Process

Instruction: You are an expert who works with theorem provers. Given some context and a question, generate the predicates and the first-order logic formula for contexts and question. Here is an example.

—

Context:
Anne is quiet. Erin is furry. Erin is green. Fiona is furry. Fiona is quiet. Fiona is red. Fiona is rough. Fiona is white. Harry is furry. Harry is quiet. Harry is white. Young people are furry. If Anne is quiet then Anne is red. Young, green people are rough. If someone is green then they are white. If someone is furry and quiet then they are white. If someone is young and white then they are rough. All red people are young.

Question:
Anne is white.

###
Predicates:
`Quiet(x)` ::: x is quiet
`Furry(x)` ::: x is furry
`Green(x)` ::: x is green
`Red(x)` ::: x is red
`Rough(x)` ::: x is rough
`White(x)` ::: x is white
`Young(x)` ::: x is young

Premises:
`Quiet(Anne)` ::: Anne is quiet.
`Furry(Erin)` ::: Erin is furry.
`Green(Erin)` ::: Erin is green.
`Furry(Fiona)` ::: Fiona is furry.
`Quiet(Fiona)` ::: Fiona is quiet.
`Red(Fiona)` ::: Fiona is red.
`Rough(Fiona)` ::: Fiona is rough.
`White(Fiona)` ::: Fiona is white.
`Furry(Harry)` ::: Harry is furry.
`Quiet(Harry)` ::: Harry is quiet.
`White(Harry)` ::: Harry is white.
$\forall x\,(\texttt{Young}(x) \rightarrow \texttt{Furry}(x))$ ::: Young people are furry.
$\texttt{Quiet(Anne)} \rightarrow \neg\texttt{Red(Anne)}$ ::: If Anne is quiet then Anne is not red.
$\forall x\,(\texttt{Young}(x) \wedge \texttt{Green}(x) \rightarrow \texttt{Rough}(x))$ ::: Young, green people are rough.
$\forall x\,(\texttt{Green}(x) \rightarrow \texttt{White}(x))$ ::: If someone is green then they are white.
$\forall x\,(\texttt{Furry}(x) \wedge \texttt{Quiet}(x) \rightarrow \neg\texttt{White}(x))$ ::: If someone is furry and quiet then they are not white.
$\forall x\,(\texttt{Young}(x) \wedge \texttt{White}(x) \rightarrow \texttt{Rough}(x))$ ::: If someone is young and white they are rough.
$\forall x\,(\texttt{Red}(x) \rightarrow \texttt{Young}(x))$ ::: All red people are young.

Conclusion:
`White(Anne)` ::: Anne is white.

—
<INPUT>
###

Figure 3: Few shot example and instruction for FOL generation of PROOFFOL using `gpt4o`.

Prompt and a Few-shot Example for ICL FOL Generation

System:
Given a set of premises and a conclusion, generate the predicates and first-order logic representation of both the premises and the conclusion.

User:
Premise:
Braden advances medical knowledge. If Braden pursues science, then he can either make a groundbreaking discovery or advance medical knowledge, but not both. Braden pursues science.
Conclusion:
Braden makes a groundbreaking discovery.
Assistant:
Predicates:
AdvancesMedicalKnowledge(x);
PursuesScience(x);
MakesDiscovery(x);
Premise_First-order:
AdvancesMedicalKnowledge(Braden) ::: Braden advances medical knowledge.;
PursuesScience(Braden) $\rightarrow$ (MakesDiscovery(Braden) $\oplus$ AdvancesMedicalKnowledge(Braden)) ::: If Braden pursues science, then he can either make a groundbreaking discovery or advance medical knowledge, but not both.;
PursuesScience(Braden) ::: Braden pursues science.
Conclusion_First-order:
MakesDiscovery(Braden) ::: Braden makes a groundbreaking discovery.

Other Few-Shot Examples

User:
Premises:
Lennon has feathers. If Lennon either eats pollen or sips sap (but not both), then he collects nectar. Lennon does not locate food. Lennon does not avoid predators. Any insect that finds shelter either avoids predators or locates food (or both). If Lennon has feathers, then he spreads seeds and flies efficiently. If Lennon hibernates during winter, then he migrates seasonally. Louisa does not avoid predators. Lennon finds shelter or migrates seasonally. If Karim has feathers, then he spreads seeds and flies efficiently. Lennon either collects nectar or spreads seeds, but not both.
Conclusion:
Lennon does not pollinate flowers.

Figure 4: Prompt for FOL Generation in an ICL setting for Instruct models.