Fine-tuning LLMs for logical deduction

Olaf Werner and Marcin Luckner

***Abstract*—Large Language Models (LLMs) have demonstrated potential for reasoning through techniques such as Chain-of-Thought (CoT) prompting. However, their reasoning capabilities often falter, especially with complex or multi-step logical deductions. This work uses symbolic Reasoning Engines to explore the fine-tuning of LLMs specifically for logical reasoning tasks. Using five reasoning datasets, we evaluate standard prompting, CoT, and logic engine-based approaches. Our fine-tuning process incorporates low-rank adaptation (LoRA) and NEFTune techniques, optimising LLMs to generate structured solutions compatible with symbolic solvers. The results indicate significant performance improvements, with fine-tuned models achieving higher accuracy across some datasets than GPT-4. This work underscores the potential of specialised training for using logic engines to enhance LLM reasoning capabilities.**

***Index Terms*—Large Language Models (LLMs), Logical Reasoning, Fine-Tuning, Symbolic Reasoning Engines, Formal Logic Translation, Neural-Symbolic AI**

I. Introduction

Large Language Model (LLM) is capable of reasoning by CoT prompting. Unfortunately, this reasoning is often imperfect and can fail, especially when reasoning can take multiple or long paths [1]. One way to remedy this and make reasoning more transparent is to use Reasoning Engines (REs) such as Pyke, Prover9, or Z3 [2]–[4]. These engines are designated tools for reasoning tasks. However, their main limitation is that the logical rules must be entered in a special format. The need to handcraft rules by experts was a major bottleneck for such expert systems. Thankfully, LLMs can substitute humans in the rule-creation process. Unfortunately, LLMs were not explicitly trained for that task, so rules are not always created perfectly. To avoid the problems of integrating REs with LLMs, some work trained LLMs to act as logic solvers [5], but this makes reasoning no longer explainable. Other works, such as [6], have followed a similar process of extracting rules and applying an RE.

A unique contribution of this work is the fine-tuning of LLM for reasoning, which was not present in previous works, i.e. [6]. This approach gives better results as LLMs were not explicitly trained to deal with the translation of natural languages to formal ones. First, we reproduce the findings of [7] using LLama-2 and Mistral LLMs. Then, we showcase our method, which involves generating solutions using a combination of LLMs and REs, filtering correct reasonings, and finetuning LLMs on them. As a result, the percentage of correctly processed tasks increased and, for some data sets, extended the results obtained by more complex LLMs like GPT.

The rest of this paper is structured as follows. Section II briefly presents related works in LLM's prompting, neuro-symbolic integration and finetuning. Section III describes our approach to prompting, used benchmarks and reasoning engines, tested LLMs and their configuration, and tuning. Section IV shows results of fine-tuning. Finally, Section V summarises the paper and proposes future works.

O. Werner and M. Luckner were with the Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland.
E-mail: olaf.werner.dokt@pw.edu.pl, mluckner@mini.pw.edu.pl.

This research was carried out with the support of the Laboratory of Bioinformatics and Computational Genomics and the High Performance Computing Center of the Faculty of Mathematics and Information Science Warsaw University of Technology.

II. Related Works

The literature defines three approaches to achieving better reasoning in LLMs that domain works fall into. It is worth noting that these approaches are not mutually exclusive and can be used together.

A. Prompting and Generation/Search Strategies

The first approach modifies the way of generating new tokens. One way to do it is to ask it to generate a reasoning trace before outputting the answer. This approach is known as Chain-of-thought (CoT). This was shown to improve reasoning in [8].

Another way is to modify ways are generating new tokens. Normally LLM generates tokens sequentially with no way to change already inputted tokens. However, possible answers can be explored more thoroughly by branching out during generation. An approach that is doing that is called Tree of Thoughts (ToT) [9]. This approach is a generalisation of CoT, and after every *thought* (longer text sequence), several paths are explored, and paths deemed unpromising by the evaluator are cut.

There are other approaches with a similar spirit, like using beam search [10] or Monte Carlo Tree Search [11]. The main idea is to demand reasoning traces and perform a search on these traces.

B. Neuro-Symbolic Integration

Neuro-symbolic reasoning systems aim to bridge the gap between neural network flexibility and symbolic systems rigour. Recent works – describing such systems as LOGIC-LM [7] and CARING [6] – have showcased how integrating symbolic solvers with LLMs can visibly enhance reasoning accuracy and reliability. By translating natural language into symbolic representations, these methods utilise external solvers to conduct precise reasoning, achieving substantial improvements over traditional LLM-based reasoning systems. The CARING, for instance, utilises Prolog to ensure deterministic reasoning processes, while LOGIC-LM integrates feedback mechanisms to iteratively refine symbolic formalisations.

An interesting approach was presented in [12], where the authors proposed a model to convert structural logic expressions

into natural language. It may seem as a reverse of the issue described in our work.

C. Fine-tuning for reasoning

Another approach is to simply fine-tune the model for logical reasoning. While fine details may be different, all of them work in the following way. The system generates solutions for tasks and evaluates them to give them scores. Next, the model is updated using scored solutions. The entire process is repeated with the improved model if necessary. The approach can be observed in several works.

In [5], the authors introduced the LOGIPT language model that directly emulated the reasoning processes of logical solvers and bypassed the parsing errors by learning to strict adherence to solver syntax and grammar. The experiment performed on PrOntoQA and ProofWriter received results similar to or better than GPT-4.

In [11], the authors enhanced the reasoning capabilities of LLMs through an iterative preference learning process inspired by the AlphaZero strategy. The authors leveraged Monte Carlo Tree Search to iteratively collect preference data, utilising the ability to break down instance-level rewards into more granular step-level signals. Evaluating various arithmetic and commonsense reasoning tasks demonstrated performance improvements over Mistral-7B.

In [13], the authors proposed a technique to iteratively leverage a small number of rationale examples and a large dataset without rationales, to bootstrap the ability to perform successively more complex reasoning. The authors showed that their system significantly improves performance on multiple datasets compared to a model fine-tuned to directly predict final answers.

III. Methods

The proposed approach, presented in Fig. 1, consists of the following elements. In the first step, data from the dataset are gathered to generate prompts for the analysed models based on the prompt templates.

The next step is the generation of examples. Using a combination of LLM and RE, the solutions are generated. The generation process consists of LLM generating a logical program for RE and using RE to get final results. The whole generation pipeline is summarised in Fig. 2.

The obtained results are filtered to create training data for the models. It is worth noting that a given model can be trained on the outputs of other models. This is especially important for the smallest models, which, as will be shown in later chapters, are unable to generate good solutions initially but, after training on outputs of bigger models, became capable of generating good solutions.

A. Prompting

The prompt is a starting instruction to the language model. In the proposed methodology, few-shot prompting is applied [14]. In this approach, the question to the model is preceded by examples of correct solutions.

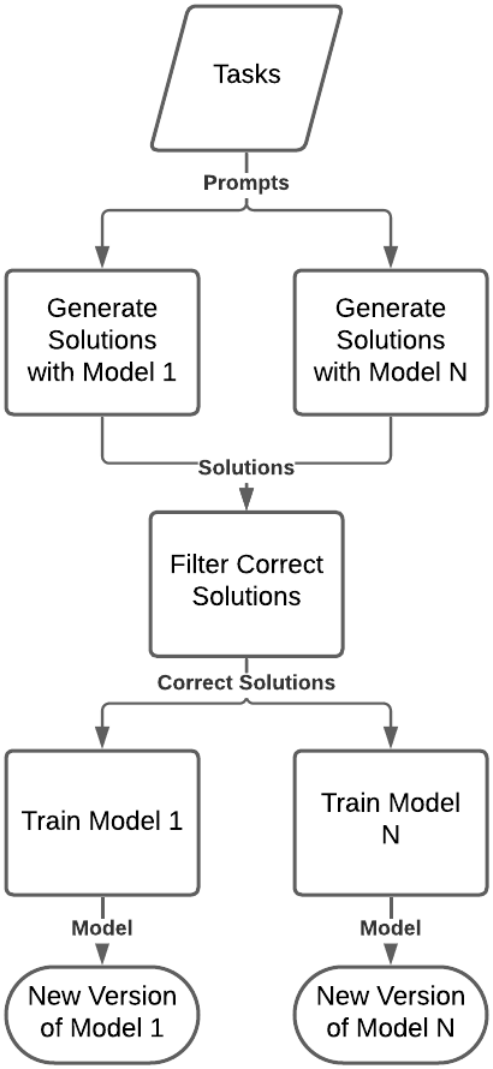


Fig. 1: Summary of proposed approach for training

Without a RE, the prompts can be addressed directly (Direct) or as a CoT. In the Direct approach, the model answers immediately, so it takes only a few tokens to write the answer. In the case of CoT prompting, the model is asked to show its reasoning towards the solution and then to answer the question. To keep the generation deterministic, the temperature parameter is set at 0. Multi-turn prompting, error correction, or more complex flows were not applied.

Direct and CoT approaches were used as baseline approaches to compare with the proposed method.

B. Datasets

The datasets examined in our experiments are the same benchmark used in [7]. The datasets differ in types of described problems and type of used RE. Details on the datasets in the benchmark are presented below.

PrOntoQA [1] is a synthetic deductive reasoning dataset. Because of its synthetic nature, we generated additional data

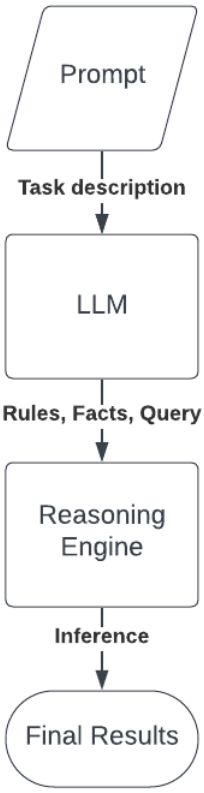


Fig. 2: Summary of proposed approach for generation

using the *Modus Ponens* mode, fictional entities, and hops of length from 2 to 4 for every 500 samples. The train set consists of 1500 examples. The test set contains 500 examples. Possible answers are True and False. Example prompt is in A.

ProofWriter [15] is a deductive reasoning dataset. The train set consists of 3000 examples. The test set contains 600 examples. Possible answers are True, False, and Unknown. Example prompt is in B.

FOLIO [16] is a dataset for logical reasoning. The train set consists of 1004 examples. The test set consists of 204 examples. Possible answers are True, False, and Uncertain. Example prompt is in D.

LogicalDeduction [17] is a logical reasoning benchmark from the BigBench. The train set consists of 1200 examples. The test set contains 300 examples. Possible answers are A, B, C, D, and E. Example prompt is in C.

AR-LSAT [18] is an analytical logic reasoning dataset from the Law School Admission Test from 1991 to 2016. The train set consists of 1585 examples. The test set contains 231 examples. Possible answers are A, B, C, D, and E. Example prompt is in E.

Because benchmarks are different kinds of logical problems (for example, deductive reasoning in PrOntoQA and Constraint Satisfaction in the case of AR-LSAT), various REs have to be used for various datasets.

C. Reasoning Engines

The method used in the proposed approach is rewriting the problem to an RE, which should solve it. The method is referred to as Logic in the rest of the article. In the Logic approach, the LLM is instructed to translate the problem into a form that the RE can analyse. The LLM performs the translation using an example of a correctly translated problem.

The following sections summarise solvers tested during the experiments. The examples for separate RE are given after [7]

1) Pyke: Pyke [2] is the expert system for deductive reasoning. Pyke creates a knowledge base based on a query and populates it with facts and rules. Then, it uses forward/backward-chaining algorithms to answer the query.

The system replaces logical tasks such as the following sentence *If the circuit is complete and the circuit has the light bulb then the light bulb is glowing.* into a logical formula such as

$$\begin{aligned} &Complete(Circuit, True) \wedge \\ &Has(Circuit, LightBulb) \rightarrow \\ &Glowing(LightBulb, True). \end{aligned}$$

According to [7], the RE could be used for PrOntoQA and ProofWriter datasets. However, we could not reliably replicate their results. The reason for this is that Pyke is using a cache file to populate its database with facts and rules. However, issues with cleaning the cache file occurred, which caused multiple problems when running it, so the results were not deterministic. Also, this particular project is not regularly maintained as the last commit was in April 2010, and finding information online regarding it was difficult. Because of these technical problems, Pyke was replaced with Prover9, which is described in the next section.

2) Prover9: Prover9 [3] is a first-order logic theorem prover. It converts FOL statements to Conjunctive Normal Form (CNF) and then performs resolution on the CNF to deduce whether a conclusion is true, false, or unknown.

The Prover9 represents the sequence *A Czech person wrote a book in 1946* as the following symbolic formula:

$$\begin{aligned} \exists x_2 \exists x_1 Czech(x_1) \wedge \\ Author(x_2, x_1) \wedge \\ Book(x_2) \wedge \\ Publish(x_2, 1946). \end{aligned}$$

Prover9 was used in our experiments on PrOntoQA, ProofWriter, and FOLIO datasets.

3) python-constraint: The python-constraint [19] is a Constraint Satisfaction Poblems (CSP) solver. The goal is to find such values of variables that none of the constraints are volatiled.

The sentence *On a shelf, there are five books. The blue book is to the right of the yellow book.* is represented by the following formula

$$blue\_book \in \{1,2,3,4,5\}$$
$$yellow\_book \in \{1,2,3,4,5\}$$
$$blue\_book > yellow\_book.$$

The python-constraint was used in our experiments on LogicalDeduction dataset.

4) Z3: Z3 [4] is a theorem prover for solving propositional satisfiability problems (SAT). It searches for such an assignment of logical values that the entire set of formulas evaluates to True.

The sentence *Xena and exactly three other technicians repair radios.* is represented by the following formula

$$repairs(Xena, radios) \wedge$$
$$Count([\mathrm{t} : technicians], \mathrm{t} \neq Xena$$
$$\wedge repairs(\mathrm{t}, radios))) == 3).$$

Z3 was used in our experiments on AR-LSAT dataset.

D. Large Language Model

The experiment was performed on LLMs from two families Llama-2 [20] and Mistral [21]. The families were selected because of their popularity. In the first cases, three models were considered Llama-2-7b-hf, Llama-2-13b-hf, and Llama-2-70b-hf. In the second case, two models were considered Mistral-7B-v0.1 and Mixtral-8x7B-v0.1 (Mixture of Expert version [22]).

1) Stopping Condition: We noticed that LLMs have trouble stopping generation properly, so generation is stopped too early or goes longer than needed. In case of going longer than needed, untrained LLM repeats the previous task while trained LLM starts generating new previously unseen tasks, which is curious behaviour, but was not investigated in more details.

On the other hand, generation was sometimes stopped abruptly before the task was completed. To circumvent this issue, we take advantage of our prompt template. Tasks are separated by special sequence \n——. Therefore, using the EOS (end of sequence) token was forbidden until the separator was seen. When the separator occurred, the generation immediately ended.

2) Generating of Training Sets: While the train data contains questions and answers, it lacks solutions in the form Logic Solvers needs. Therefore, the tested LLMs generated training data with correct formulations. Among the formulations, only correct solutions were selected. The solution is proper when the formula is generated in a form that can be compiled using the RE and when the coding is correct according to the answer.

Because several LLMs were used to answer the same problem, the duplicated answers were removed. However, even after deduplication, multiple answers can exist to the same task. This is because the differences in answers are stylistic. For example, some models write a bit different comments in their reasoning. This is not an issue, just effective number of epochs can be different as the same task can appear multiple times.

TABLE I: Datasets summary: the original size, the total number of processed examples, and the number of unique tasks

Dataset	Size	Train examples	Unique tasks
PrOntoQA	1500	1969	1202
ProofWriter	3000	2236	1514
FOLIO	1004	1101	552
LogicalDeduction	1200	1682	826
AR-LSAT	1585	123	101

A similar procedure was proposed in [13]. However, we have not yet regenerated the answers with improved models. The summary of created train datasets is presented in Tab. I.

The table presents the original size of each dataset, the total number of processed examples used to train the models, and the number of unique tasks solved by the models.

3) 8-bit Quantization: Usually, the parameters of neural networks are stored in FP-16 or FP-32 format. Numbers 16 and 32 refer to the number of bites used to start a floating point number. It is possible to load them in INT-8 format to reduce memory usage, but this can degrade their quality. In [23], this issue is discussed and worked around by separating matrix operations into two parts. More important parameters are kept in 16-bit format while the rest is transformed into 8-bit. The important features are called outliers. All hidden states with absolute values greater than 6 are considered outliers in our work. The same threshold was suggested in [23].

During the matrix multiplication, the calculations are split into two matrices. All values that would be multiplied with outliers stay in FP-16 format. Other values get quantised in INT-8 format. Then, the two separate matrix multiplications, at different precisions, are performed and summarised together to obtain the final results.

E. Tuning

1) Low-Rank Adaptation: For the fine-tuning, Low-Rank Adaptation (LoRA) [24] was used. The main idea of this method is to represent the weight matrix by its low-rank decomposition. When weight matrix $W_0 \in \mathbb{R}^{d\times k}$ is tunned, its update is represented by decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d,k)$. During the training, W_0 is not updated, while A and B contain trainable parameters. For $h = W_0 x$, the final modified forward pass is:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

The method vastly reduces the number of tunable parameters. In our work, parameter α was set to r as in the original paper [24]. The bias was not targeted, and the dropout was 0.05. According to the authors' directions, the tuning covered all linear layers to obtain the best results.

2) NEFTune: NEFTune [25] is a technique for training LLMs. A random noise vector is added to the embeddings, where the noise is drawn from independent and identically distributed uniform values in the range $[-1,1]$. Next, the entire noise vector is scaled by the factor $\frac{\alpha}{\sqrt{Ld}}$, where L is the

TABLE II: Accuracy of direct prompting (D), chain-of-thought prompting (CoT), using logic-engine (L) on the reasoning datasets, encoded as follows: PrOntoQA (P), ProofWriter (PW), FOLIO (F), LogicalDeduction (LD), and AR-LSAT (AL)

Data	Llama-2-7b			Llama-2-13b			Llama-2-70b			Mistral-7B			Mixtral-8x7B		
	D	CoT	L	D	CoT	L	D	CoT	L	D	CoT	L	D	CoT	L
P	25.8	**31.8**	1.0	6.0	4.4	**17.8**	5.0	44.4	**70.6**	**55.8**	40.0	35.2	58.0	41.8	**99.2**
PW	3.6	**24.8**	6.5	4.3	4.9	**6.2**	3.8	30.0	**52.2**	36.8	34.6	**43.5**	42.3	35.6	**73.0**
F	3.4	**29.4**	11.3	5.3	6.3	**22.5**	4.9	**32.8**	27.5	**50.9**	34.8	19.1	**49.5**	34.8	30.9
L	**21.3**	17.0	18.3	5.6	**12.0**	2.0	30.6	10.0	**44.7**	32.0	23.3	**50.0**	**39.6**	28.3	29.0
AL	**15.1**	12.5	0.9	3.0	**8.6**	2.6	1.7	**11.6**	3.03	**21.6**	13.2	3.5	**24.6**	12.5	3.0

sequence length, d is the embedding dimension, and α is a tunable parameter.

In our work, we test the value of α of 0, the equivalent of turning the NEFTune off, and 5, the default value proposed in the implementation.

3) Supervised Fine Tuning: The supervised fine-tuning task predicts the next token based on the previous ones. The AdamW optimiser can be used for that [26].

All runs presented in this paper were started with the following hyperparameters that correspond to the default parameters used by the Transformers library.

- learning_rate (5e-5) — The initial learning rate for AdamW optimiser.
- weight_decay (0) — The weight decay to apply
- adam_beta1 (0.9) — The beta1 hyperparameter for the AdamW optimizer.
- adam_beta2 (0.999) — The beta2 hyperparameter for the AdamW optimizer.
- adam_epsilon (1e-8) — The epsilon hyperparameter for the AdamW optimiser.
- batch_size (automatic depending on available hardware and used model) - The batch size
- epoch (1) - number of epochs

The size of the validation set was fixed at 15 per cent of the train data. We set the number of epochs to 1 as multiple solutions can appear in the same task as multiple models generate them.

4) Few-shot vs Zero-shot: Few-shot and Zero-shot learning are paradigms of learning LLMs (but not only) with little or no labelled data, i.e. [27], [28]. Their usefulness depends on a specific task, and we could not find suitable research whenever a few shot examples should be kept or removed when performing training for logical deduction. It is also unclear if a few shot examples should be kept after finetuning. Because of these doubts, all training and generation routine combinations were applied in the experiments.

IV. Results

A. Baseline results

The baseline results obtained by Llama and Mistral models were collected in Tab.II. The presented results were obtained without fine-tuning but for three aftermentioned approaches: Direct, CoT, and Logic.

In many cases, the straightforward approach – the Direct prompt – works better than CoT. The Logic approach can improve the results, but it cannot always. Most noteworthy is that Mixtral-8x7B-v0.1 managed to get 99.2 per cent accuracy for the ProntoQA dataset and 73 per cent for the ProofWriter dataset. Overall, not instruction-tuned LLMs can not use reasoning with CoT, but the biggest LLMs can use Logic engines semi-reliably.

B. Fine Tuning Experiments

The fine-tuning was performed on the following parameters:

- Zero-shot training $\in \{On, Off\}$: Is zero-shot used for training.
- Zero-shot generation $\in \{On, Off\}$: Is zero-shot used for a generation.
- NEFTune $\alpha \in \{0, 5\}$: NEFTune α parameter for scaling noise.
- LoRA $r \in \{8, 16, 32\}$: LoRA r parameter which determines rank of used matrices.

In the initial step, a full hyperparameter search was performed on all datasets but only one model of lower complexity (Llama-2-7b-hf).

1) Full Fine Tuning Results: The results of full fine-tuning created a 5×24 matrix with rows representing the analysed datasets and columns being unique combinations of hyperparameters.

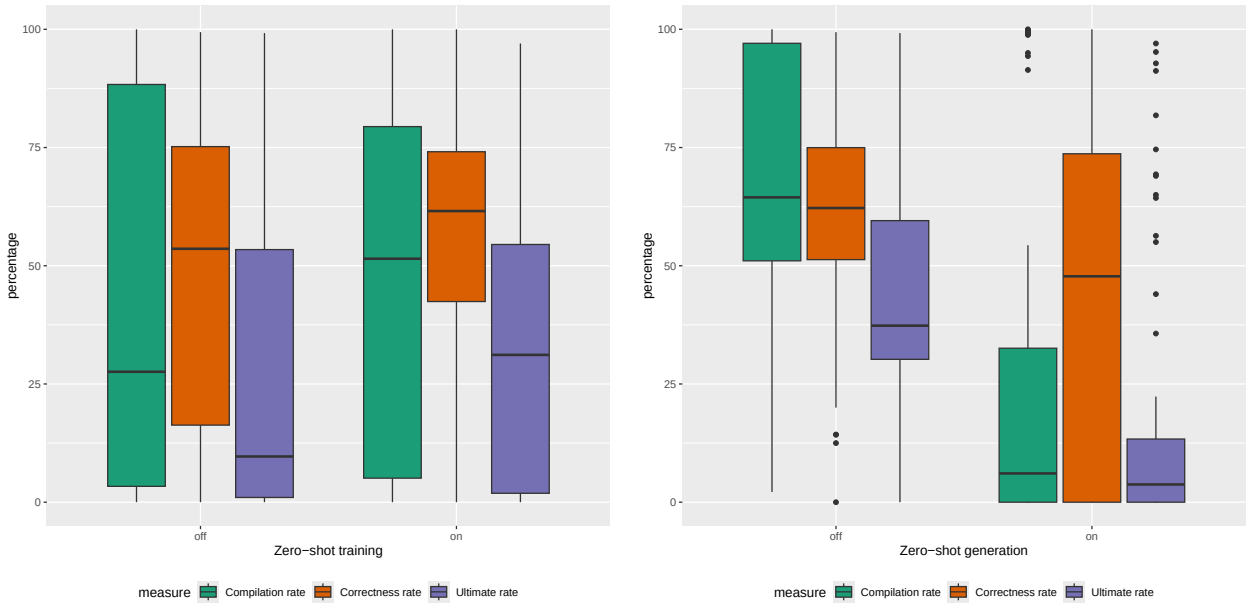
To find if there is a significant difference between results obtained for various hyperparameters, a Friedman test [29] was performed. After that, the Nemenyi post-hoc test [30] was performed. Nemenyi test is used to determine which groups are significantly different from each other, while Friedman checks globally for the difference. These are ranked tests, and we determine ranks based on the accuracies of the model on a given dataset.

The value of the Friedman test was 72.87, and the p-value was 4.33e-07. Therefore, the null hypothesis that hyperparameter sets have equal rank was rejected.

Next, the following set of hyperparameters with the lowest average rank being equal to 4.3. was selected:

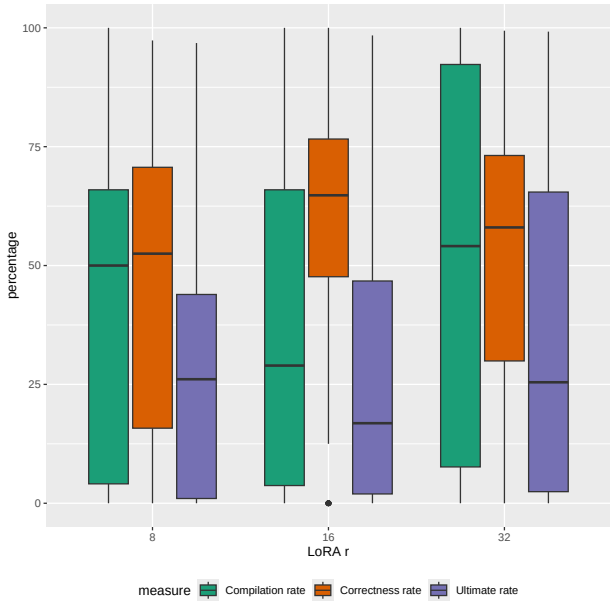
- Zero-shot training: Off
- Zero-shot generation: Off
- NEFTune α: 5
- LoRA r: 32

The results of the Nemenyi post-hoc test and comparison of p-values for the best set of hyperparameters with the rest are presented in Tab. III. The lower the average rank, the better. When the p-value is greater than 0.05, that means the given

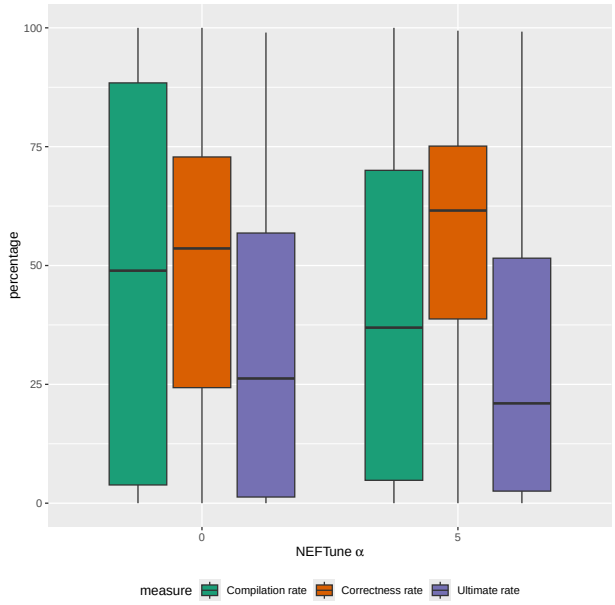


(a) Zero-shot training

(b) Zero-shot generation



(c) LoRA r



(d) NEFTune α

Fig. 3: Influence of parameters in Llama-7b fine-tuning experiments on compilation rate and accuracies of compiled expressions

set of hyperparameters is not significantly worse compared to the best set of hyperparameters. As we can see, there are a lot of hyperparameters with high p-values, so our training is not too sensitive to the chosen set of hyperparameters.

Analysing Tab. III and Fig. 3, which presents the percentage of compiled and correct causes from the testing set, several important insights can be drawn.

Tab. III shows that the worst results were obtained when the network was trained with few-shot examples in training (Zero-shot training: False) and examples removed during the generation (Zero-shot generation: On). On the other hand, the best set of hyperparameters is not dominating over other options. That shows that the model is not too sensitive regarding the hyperparameter selection. It is desirable because the same parameters will be applied in other networks (more complex LLama model and Mistral).

Another interesting insight is that few-shot examples help even if the model was trained without them (see Fig. 3a). However, we observed that if the model has mastered the task, like in the case of ProntoQA, they are no longer helpful. Fig. 3b shows that also, in the case of generation, the zero-shot should be off. Fig. 3c shows that the bigger LoRA r, the better, but not necessary in the context of the mean value. Finally, Fig. 3d shows that the NeFTune does not significantly change results.

TABLE III: Nemenyi post-hoc test results and p-value from comparing to the best set of hyperparameters consisting of Zero-shot training and generating, LoRA r, and NEFTune α

LoRA r	ZS train.	NEFTune α	ZS gen.	Nemenyi p-value	Ave. rank
8	Off	0	Off	1.00	10.10
			On	0.03	21.20
		5	Off	1.00	9.30
			On	0.05	20.70
	On	0	Off	0.99	12.10
			On	0.91	13.70
		5	Off	0.99	11.90
			On	0.81	14.60
16	Off	0	Off	1.00	6.40
			On	0.03	21.10
		5	Off	1.00	6.40
			On	0.03	21.30
	On	0	Off	1.00	9.50
			On	0.98	12.50
		5	Off	1.00	10.10
			On	0.96	13.00
32	**Off**	0	Off	1.00	4.90
			On	0.14	19.10
		5	**Off**	1.00	4.30
			On	0.09	19.70
	On	0	Off	1.00	5.90
			On	0.91	13.70
		5	Off	1.00	7.10
			On	1.00	11.40

C. Global Fine Tuning Results

Tab. IV shows the final results obtained after fine-tuning by LLama and Mistral models, respectively. Fig. 4 compares the results with other models: gpt-3.5-davinci and gpt-4. We take the results from [7]

First, apart from the AR-LSAT dataset, there is a visible improvement in accuracy for all models. The AR-LSAT dataset, see Fig. 4a, is worth noticing as all models perform badly. This was caused by the fact that we had a dataset too small for training, consisting of only 101 examples. This was the hardest dataset for even other commercial models, gpt-3.5-davinci and gpt-4. So we had the fewest samples on the hardest dataset when the most were needed, and untrained models could not generate samples for training. The work [13] also highlighted similar problems with bootstrapping.

Mistral-7B-v0.1 was a better model than Llama-2-7b-hf and its own MoE variant, Mixtral-8x7B-v0.1, on all datasets apart from AR-LSAT and slightly worse on ProntoQA (see Fig. 4d).

This could be caused by the fact that Mixtral-8x7B-v0.1 is an MoE model, and the influence of parameter scaling is not so transparent in the case of the Llama-2 family on ProofWriter and LogicalDeduction datasets (see Fig. 4e and Fig. 4c).

A strange anomaly can be observed for the Llama-2-13b-hf model in the case of the ProntoQA dataset (see Fig. 4d). This model's compiled accuracy is 97.11%, but the compilation rate is only 41.6%. On other datasets, this model performs better than its smaller variant Llama-2-7b-hf. After investigations, we encounter extreme forms of hallucinations. Sometimes, the model, instead of doing its task, starts to make a login webpage or many hashtags. We have no explanation for this behaviour. Example output is in Appendix A. We also do not observe this behaviour for the bigger variant Llama-2-70b-hf.

The most satisfying result is that after training, our models exceeded gpt-3.5-davinci and even gpt-4 on ProofWriter and ProntoQA datasets (see Fig. 4e and Fig. 4d). This showcases the value of our method that further specialised training in tool use on much smaller models can make them more accurate than larger models.

Unfortunately, for AR-LSAT, FOLIO, and LogicalDeduction datasets, we get worse results than gpt-4. On LogicalDeduction (see Fig. 4c) all models apart from Llama-2-7b-hf are better than gpt-3.5-davinci.

V. Conclusions and Future Work

This research presents an effective methodology for fine-tuning LLMs using symbolic solvers to perform logical reasoning tasks.

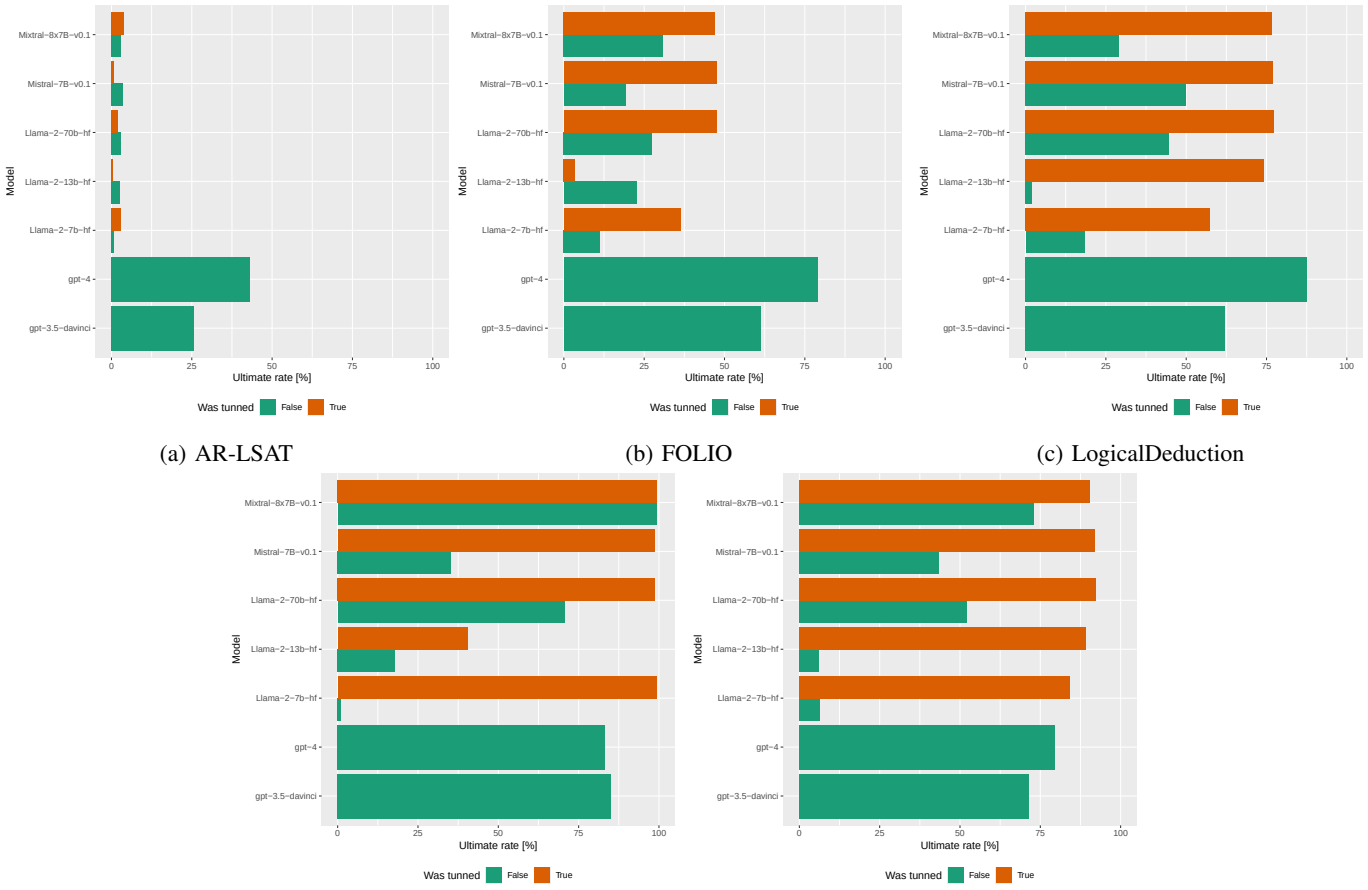
The paper demonstrates improved reasoning accuracy on benchmark datasets by addressing inherent limitations of existing reasoning techniques, such as CoT prompting, and integrating tools like Prover9 and Z3. The presented experiments reveal the value of low-rank adaptation in boosting LLM performance without excessive computational overhead. The importance of using the correct prompt structure during fine-tuning and later generation is also shown. However, challenges such as hallucination and abrupt stopping in generation persist, indicating areas for future exploration. Ultimately, this work advances the potential of LLMs for transparent and explainable AI applications, paving the way for further innovations in logic-driven reasoning systems.

While writing this work, the authors have preliminary tested some of the possibilities, but not to the extent that they can be included in this paper. This section should be treated as suggestions based on intuitions.

It is possible to leverage more computing during inference by using beam search [10] to improve results, especially compilation. However, initial gains seem to diminish while scaling beam numbers further quickly. Also, while LLMs can produce solutions to many problems, there is no guarantee that they will perform the task correctly; having an additional model to check if the solution is correct and give us the calibrated probability of being correct would be useful. Finally, An iterative approach can be applied – like in [13] – instead of generating programs for the train set only once.

The fact that each puzzle type required a different solver made it harder to check for generalisation capabilities to different tasks and craft prompts with examples. More robust frameworks should be adopted; we propose using mathematical proof checkers like LEAN or CoQ as they are, in principle, the most capable systems.

The presented tests focused on five different reasoning benchmarks, but this set can be extended, i.e. using LogiQA 2.0 [31]. Including more benchmarks for training and testing



(a) AR-LSAT (b) FOLIO (c) LogicalDeduction

(d) ProntoQA (e) ProofWriter

Fig. 4: Final results comparing accuracies of multiple models. Results of GPT family of models taken from [7]

TABLE IV: Summary of logic engine results after fine-tuning. *Ult* is overall accuracy, *Com* is the percentage of compiled programs, *Cor* is the accuracy of programs that complied. The data sets are encoded as follows: PrOntoQA (P), ProofWriter (PW), FOLIO (F), LogicalDeduction (LD), and AR-LSAT (AL).

Data	Llama-2-7b			Llama-2-13b			Llama-2-70b			Mistral-7B			Mixtral-8x7B		
	Ult	Com	Cor	Ult	Com	Cor	Ult	Com	Cor	Ult	Com	Cor	Ult	Com	Cor
P	99.2	99.4	99.8	40.4	97.1	41.6	98.8	99.0	99.8	98.6	100.0	98.6	99.4	100.0	99.4
PW	84.3	86.8	97.2	89.2	90.2	98.8	92.3	92.9	99.3	91.8	92.6	99.2	90.5	93.3	97.0
F	36.3	54.8	66.2	3.4	38.9	8.8	47.5	57.1	83.3	47.5	61.4	77.5	47.1	60.0	78.4
LD	57.3	63.6	90.3	74.3	74.3	100.0	77.3	77.3	100.0	77.0	77.0	100.0	76.7	76.7	100.0
AL	3.03	36.8	8.2	0.4	33.3	1.3	2.2	14.3	15.2	0.9	13.3	6.5	3.9	30.0	13.0

purposes would be beneficial; however, first more universal symbolic solver should be chosen to make an expansion easier. Also, having easier task variants could make fine-tuning models easier as they would get competencies on easier tasks, so bootstrapping processes would be better. Similarly, the field of LLMs is developing rapidly, and new model families like LLama-3 [32] were released so a better performance can be obtained simply by scaling and new model families can be added to the comparison.

REFERENCES

[1] A. Saparov and H. He, "Language models are greedy reasoners: A systematic formal analysis of chain-of-thought," 2023.

[2] B. Frederiksen, "Applying expert system technology to code reuse with pyke," *PyCon: Chicago*, 2008.

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

[4] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.

[5] J. Feng, R. Xu, J. Hao, H. Sharma, Y. Shen, D. Zhao, and W. Chen, "Language models can be logical solvers," 2023.

[6] S. Yang, X. Li, L. Cui, L. Bing, and W. Lam, "Neuro-symbolic integration brings causal and reliable reasoning proofs," *ArXiv*, vol. abs/2311.09802, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265221449

[7] L. Pan, A. Albalak, X. Wang, and W. Wang, "Logic-LM: Empowering large language models with symbolic solvers for faithful logical

reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3806–3824. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.248

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[9] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.10601

[10] A. Graves, "Sequence transduction with recurrent neural networks," 2012. [Online]. Available: https://arxiv.org/abs/1211.3711

[11] Y. Xie, A. Goyal, W. Zheng, M.-Y. Kan, T. P. Lillicrap, K. Kawaguchi, and M. Shieh, "Monte carlo tree search boosts reasoning via iterative preference learning," 2024. [Online]. Available: https://arxiv.org/abs/2405.00451

[12] X. Wu, Y. Cai, Z. Lian, H.-f. Leung, and T. Wang, "Generating natural language from logic expressions with structural representation," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 31, p. 1499–1510, Apr. 2023. [Online]. Available: https://doi.org/10.1109/TASLP.2023.3263784

[13] E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman, "Star: Bootstrapping reasoning with reasoning," 2022. [Online]. Available: https://arxiv.org/abs/2203.14465

[14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[15] O. Tafjord, B. Dalvi, and P. Clark, "ProofWriter: Generating implications, proofs, and abductive statements over natural language," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 3621–3634. [Online]. Available: https://aclanthology.org/2021.findings-acl.317

[16] S. Han, H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, L. Benson, L. Sun, E. Zubova, Y. Qiao, M. Burtell, D. Peng, J. Fan, Y. Liu, B. Wong, M. Sailor, A. Ni, L. Nan, J. Kasai, T. Yu, R. Zhang, S. Joty, A. R. Fabbri, W. Kryscinski, X. V. Lin, C. Xiong, and D. Radev, "Folio: Natural language reasoning with first-order logic," 2022.

[17] S. et al., "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," 2023.

[18] W. Zhong, S. Wang, D. Tang, Z. Xu, D. Guo, Y. Chen, J. Wang, J. Yin, M. Zhou, and N. Duan, "Analytical reasoning of text," in *Findings of the Association for Computational Linguistics: NAACL 2022*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 2306–2319. [Online]. Available: https://aclanthology.org/2022.findings-naacl.177

[19] G. Niemeyer and S. Celles, "python-constraint 1.4.0," 2024, accessed: 2024-04-22. [Online]. Available: https://pypi.org/project/python-constraint/

[20] H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. M. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. S. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. V. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *ArXiv*, vol. abs/2307.09288, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259950998

[21] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[22] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of experts," 2024. [Online]. Available: https://arxiv.org/abs/2401.04088

[23] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," 2022. [Online]. Available: https://arxiv.org/abs/2208.07339

[24] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[25] N. Jain, P. yeh Chiang, Y. Wen, J. Kirchenbauer, H.-M. Chu, G. Somepalli, B. R. Bartoldson, B. Kailkhura, A. Schwarzschild, A. Saha, M. Goldblum, J. Geiping, and T. Goldstein, "Neftune: Noisy embeddings improve instruction finetuning," 2023. [Online]. Available: https://arxiv.org/abs/2310.05914

[26] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[27] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[28] S. Kang, J. Yoon, and S. Yoo, "Large language models are few-shot testers: Exploring llm-based general bug reproduction," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 2312–2323.

[29] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522

[30] P. B. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Princeton University, 1963.

[31] H. Liu, J. Liu, L. Cui, Z. Teng, N. Duan, M. Zhou, and Y. Zhang, "Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 31, p. 2947–2962, Jul. 2023. [Online]. Available: https://doi.org/10.1109/TASLP.2023.3293046

[32] A. G. et al., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

APPENDIX

A. Prompt for ProntoQA dataset

```
Given a problem description and a question. The task is to parse the problem and
       the question into first-order logic formulas.
The grammar of the first-order logic formula is defined as follows:
1) logical conjunction of expr1 and expr2: expr1 ∧ expr2
2) logical disjunction of expr1 and expr2: expr1 ∨ expr2
3) logical exclusive disjunction of expr1 and expr2: expr1 ⊕ expr2
4) logical negation of expr1: ¬expr1
5) expr1 implies expr2: expr1 → expr2
6) expr1 if and only if expr2: expr1 ↔ expr2
7) logical universal quantification: ∀x
8) logical existential quantification: ∃x
------
Problem:
Each jompus is fruity. Every jompus is a wumpus. Every wumpus is not transparent.
       Wumpuses are tumpuses. Tumpuses are mean. Tumpuses are vumpuses. Every vumpus
        is cold. Each vumpus is a yumpus. Yumpuses are orange. Yumpuses are numpuses
       . Numpuses are dull. Each numpus is a dumpus. Every dumpus is not shy.
       Impuses are shy. Dumpuses are rompuses. Each rompus is liquid. Rompuses are
       zumpuses. Alex is a tumpus.
Question:
True or false: Alex is not shy.
###
Predicates:
Jompus(x) ::: x is a jompus
Fruity(x) ::: x is fruity
Wumpus(x) ::: x is a wumpus
Transparent(x) ::: x is transparent
Tumpus(x) ::: x is a tumpus
Mean(x) ::: x is mean
Vumpus(x) ::: x is a vumpus
Cold(x) ::: x is cold
Yumpus(x) ::: x is a yumpus
Orange(x) ::: x is orange
Numpus(x) ::: x is a numpus
Dull(x) ::: x is dull
Dumpus(x) ::: x is a dumpus
Shy(x) ::: x is shy
Impus(x) ::: x is an impus
Rompus(x) ::: x is a rompus
Liquid(x) ::: x is liquid
Zumpus(x) ::: x is a zumpus
Premises:
∀x (Jompus(x) → Fruity(x)) ::: Each jompus is fruity.
∀x (Jompus(x) → Wumpus(x)) ::: Every jompus is a wumpus.
∀x (Wumpus(x) → ¬Transparent(x)) ::: Every wumpus is not transparent.
∀x (Wumpus(x) → Tumpus(x)) ::: Wumpuses are tumpuses.
∀x (Tumpus(x) → Mean(x)) ::: Tumpuses are mean.
∀x (Tumpus(x) → Vumpus(x)) ::: Tumpuses are vumpuses.
∀x (Vumpus(x) → Cold(x)) ::: Every vumpus is cold.
∀x (Vumpus(x) → Yumpus(x)) ::: Each vumpus is a yumpus.
∀x (Yumpus(x) → Orange(x)) ::: Yumpuses are orange.
∀x (Yumpus(x) → Numpus(x)) ::: Yumpuses are numpuses.
∀x (Numpus(x) → Dull(x)) ::: Numpuses are dull.
∀x (Numpus(x) → Dumpus(x)) ::: Each numpus is a dumpus.
∀x (Dumpus(x) → ¬Shy(x)) ::: Every dumpus is not shy.
∀x (Impus(x) → Shy(x)) ::: Impuses are shy.
∀x (Dumpus(x) → Rompus(x)) ::: Dumpuses are rompuses.
∀x (Rompus(x) → Liquid(x)) ::: Each rompus is liquid.
∀x (Rompus(x) → Zumpus(x)) ::: Rompuses are zumpuses.
Tumpus(alex) ::: Alex is a tumpus.
Conclusion:
¬Shy(alex) ::: Alex is not shy.
------
Problem:
[[PROBLEM]]
Question:
[[QUESTION]]
###
```

B. Prompt for ProofWriter dataset

```
Given a problem description and a question. The task is to parse the problem and
       the question into first-order logic formulars.
The grammar of the first-order logic formular is defined as follows:
1) logical conjunction of expr1 and expr2: expr1 ∧ expr2
2) logical disjunction of expr1 and expr2: expr1 ∨ expr2
3) logical exclusive disjunction of expr1 and expr2: expr1 ⊕ expr2
4) logical negation of expr1: ¬expr1
5) expr1 implies expr2: expr1 → expr2
6) expr1 if and only if expr2: expr1 ↔ expr2
7) logical universal quantification: ∀x
8) logical existential quantification: ∃x
------
Problem:
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:
Based on the above information, is the following statement true, false, or unknown?
        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)
Furry(erin) ∧ Green(erin)
Furry(fiona) ∧ Quiet(fiona) ∧ Red(fiona) ∧ Rough(fiona) ∧ White(fiona)
Furry(harry) ∧ Quiet(harry) ∧ White(harry)
∀x (Young(x) → Furry(x))
Quiet(anne) → Red(anne)
∀x ((Young(x) ∧ Green(x)) → Rough(x))
∀x (Green(x) → White(x))
∀x ((Furry(x) ∧ Quiet(x)) → White(x))
∀x ((Young(x) ∧ White(x)) → Rough(x))
∀x (Red(x) → Young(x))
Conclusion:
White(anne) ::: Anne is white.
------
Problem:
[[PROBLEM]]
Question:
[[QUESTION]]
###
```

C. Prompt for LogicalDeduction dataset

```
Task Description: You are given a problem description. The task is to parse the
       problem as a constraint satisfaction problem, defining the domain, variables,
        and contraints.
------
Problem:
The following paragraphs each describe a set of three objects arranged in a fixed
       order. The statements are logically consistent within each paragraph.\n\nIn
       an antique car show, there are three vehicles: a station wagon, a convertible
       , and a minivan. The station wagon is the oldest. The minivan is newer than
       the convertible.
Question:
Which of the following is true?
Options:
A) The station wagon is the second-newest.
B) The convertible is the second-newest.
C) The minivan is the second-newest.
###
Domain:
1: oldest
3: newest
Variables:
station_wagon [IN] [1, 2, 3]
convertible [IN] [1, 2, 3]
minivan [IN] [1, 2, 3]
Constraints:
station_wagon == 1 ::: The station wagon is the oldest.
minivan > convertible ::: The minivan is newer than the convertible.
AllDifferentConstraint([station_wagon, convertible, minivan]) ::: All vehicles have
        different values.
Query:
A) station_wagon == 2 ::: The station wagon is the second-newest.
B) convertible == 2 ::: The convertible is the second-newest.
C) minivan == 2 ::: The minivan is the second-newest.
------
Problem:
The following paragraphs each describe a set of five objects arranged in a fixed
       order. The statements are logically consistent within each paragraph.\n\nOn a
        branch, there are five birds: a quail, an owl, a raven, a falcon, and a
       robin. The owl is the leftmost. The robin is to the left of the raven. The
       quail is the rightmost. The raven is the third from the left.
Question:
Which of the following is true?
Options:
A) The quail is the rightmost.
B) The owl is the rightmost.
C) The raven is the rightmost.
D) The falcon is the rightmost.
E) The robin is the rightmost.
###
Domain:
1: leftmost
5: rightmost
Variables:
quail [IN] [1, 2, 3, 4, 5]
owl [IN] [1, 2, 3, 4, 5]
raven [IN] [1, 2, 3, 4, 5]
falcon [IN] [1, 2, 3, 4, 5]
robin [IN] [1, 2, 3, 4, 5]
Constraints:
owl == 1 ::: The owl is the leftmost.
robin < raven ::: The robin is to the left of the raven.
quail == 5 ::: The quail is the rightmost.
raven == 3 ::: The raven is the third from the left.
AllDifferentConstraint([quail, owl, raven, falcon, robin]) ::: All birds have
       different values.
Query:
A) quail == 5 ::: The quail is the rightmost.
B) owl == 5 ::: The owl is the rightmost.
C) raven == 5 ::: The raven is the rightmost.
D) falcon == 5 ::: The falcon is the rightmost.
E) robin == 5 ::: The robin is the rightmost.
------
Problem:
[[PROBLEM]]
Question:
[[QUESTION]]
Options:
[[CHOICES]]
###
```

D. Prompt for FOLIO dataset

```
Given a problem description and a question. The task is to parse the problem and
       the question into first-order logic formulars.
The grammar of the first-order logic formular is defined as follows:
1) logical conjunction of expr1 and expr2: expr1 ∧ expr2
```

```
2) logical disjunction of expr1 and expr2: expr1 ∨ expr2
3) logical exclusive disjunction of expr1 and expr2: expr1 ⊕ expr2
4) logical negation of expr1: ¬expr1
5) expr1 implies expr2: expr1 → expr2
6) expr1 if and only if expr2: expr1 ↔ expr2
7) logical universal quantification: ∀x
8) logical existential quantification: ∃x
------
Problem:
All people who regularly drink coffee are dependent on caffeine. People either
       regularly drink coffee or joke about being addicted to caffeine. No one who
       jokes about being addicted to caffeine is unaware that caffeine is a drug.
       Rina is either a student and unaware that caffeine is a drug, or neither a
       student nor unaware that caffeine is a drug. If Rina is not a person
       dependent on caffeine and a student, then Rina is either a person dependent
       on caffeine and a student, or neither a person dependent on caffeine nor a
       student.
Question:
Based on the above information, is the following statement true, false, or
       uncertain? Rina is either a person who jokes about being addicted to caffeine
        or is unaware that caffeine is a drug.
Based on the above information, is the following statement true, false, or
       uncertain? If Rina is either a person who jokes about being addicted to
       caffeine and a person who is unaware that caffeine is a drug, or neither a
       person who jokes about being addicted to caffeine nor a person who is unaware
        that caffeine is a drug, then Rina jokes about being addicted to caffeine
       and regularly drinks coffee.
###
Predicates:
Dependent(x) ::: x is a person dependent on caffeine.
Drinks(x) ::: x regularly drinks coffee.
Jokes(x) ::: x jokes about being addicted to caffeine.
Unaware(x) ::: x is unaware that caffeine is a drug.
Student(x) ::: x is a student.
Premises:
∀x (Drinks(x) → Dependent(x)) ::: All people who regularly drink coffee are
       dependent on caffeine.
∀x (Drinks(x) ⊕ Jokes(x)) ::: People either regularly drink coffee or joke about
       being addicted to caffeine.
∀x (Jokes(x) → ¬Unaware(x)) ::: No one who jokes about being addicted to caffeine
       is unaware that caffeine is a drug.
(Student(rina) ∧ Unaware(rina)) ⊕ ¬(Student(rina) ∨ Unaware(rina)) ::: Rina is
       either a student and unaware that caffeine is a drug, or neither a student
       nor unaware that caffeine is a drug.
¬(Dependent(rina) ∧ Student(rina)) → (Dependent(rina) ∧ Student(rina)) ⊕ ¬(
       Dependent(rina) ∨ Student(rina)) ::: If Rina is not a person dependent on
       caffeine and a student, then Rina is either a person dependent on caffeine
       and a student, or neither a person dependent on caffeine nor a student.
Conclusion:
Jokes(rina) ⊕ Unaware(rina) ::: Rina is either a person who jokes about being
       addicted to caffeine or is unaware that caffeine is a drug.
((Jokes(rina) ∧ Unaware(rina)) ⊕ ¬(Jokes(rina) ∨ Unaware(rina))) → (Jokes(rina) ∧
       Drinks(rina)) ::: If Rina is either a person who jokes about being addicted
       to caffeine and a person who is unaware that caffeine is a drug, or neither a
        person who jokes about being addicted to caffeine nor a person who is
       unaware that caffeine is a drug, then Rina jokes about being addicted to
       caffeine and regularly drinks coffee.
------
Problem:
Miroslav Venhoda was a Czech choral conductor who specialized in the performance of
        Renaissance and Baroque music. Any choral conductor is a musician. Some
       musicians love music. Miroslav Venhoda published a book in 1946 called Method
        of Studying Gregorian Chant.
Question:
Based on the above information, is the following statement true, false, or
       uncertain? Miroslav Venhoda loved music.
Based on the above information, is the following statement true, false, or
       uncertain? A Czech person wrote a book in 1946.
Based on the above information, is the following statement true, false, or
       uncertain? No choral conductor specialized in the performance of Renaissance.
###
Predicates:
Czech(x) ::: x is a Czech person.
ChoralConductor(x) ::: x is a choral conductor.
Musician(x) ::: x is a musician.
Love(x, y) ::: x loves y.
Author(x, y) ::: x is the author of y.
Book(x) ::: x is a book.
Publish(x, y) ::: x is published in year y.
Specialize(x, y) ::: x specializes in y.
Premises:
Czech(miroslav) ∧ ChoralConductor(miroslav) ∧ Specialize(miroslav, renaissance) ∧
       Specialize(miroslav, baroque) ::: Miroslav Venhoda was a Czech choral
       conductor who specialized in the performance of Renaissance and Baroque music
       .
∀x (ChoralConductor(x) → Musician(x)) ::: Any choral conductor is a musician.
∃x (Musician(x) ∧ Love(x, music)) ::: Some musicians love music.
Book(methodOfStudyingGregorianChant) ∧ Author(miroslav,
       methodOfStudyingGregorianChant) ∧ Publish(methodOfStudyingGregorianChant,
       year1946) ::: Miroslav Venhoda published a book in 1946 called Method of
       Studying Gregorian Chant.
Conclusion:
Love(miroslav, music) ::: Miroslav Venhoda loved music.
∃y ∃x (Czech(x) ∧ Author(x, y) ∧ Book(y) ∧ Publish(y, year1946)) ::: A Czech person
        wrote a book in 1946.
¬∃x (ChoralConductor(x) ∧ Specialize(x, renaissance)) ::: No choral conductor
       specialized in the performance of Renaissance.
------
Problem:
[[PROBLEM]]
Question:
[[QUESTION]]
###
```

E. Prompt for AR-LSAT dataset

```
Given a problem description and a question. The task is to formulate the problem as
        a logic program, consisting three parts: Declarations, Constraints, and
       Options.
Declarations: Declare the variables and functions.
Constraints: Write the constraints in the problem description as logic formulas.
Options: Write the options in the question as logic formulas.
------
Problem:
On Tuesday Vladimir and Wendy each eat exactly four separate meals: breakfast,
       lunch, dinner, and a snack. The following is all that is known about what
       they eat during that day: At no meal does Vladimir eat the same kind of food
       as Wendy. Neither of them eats the same kind of food more than once during
       the day. For breakfast, each eats exactly one of the following: hot cakes,
       poached eggs, or omelet. For lunch, each eats exactly one of the following:
       fish, hot cakes, macaroni, or omelet. For dinner, each eats exactly one of
       the following: fish, hot cakes, macaroni, or omelet. For a snack, each eats
       exactly one of the following: fish or omelet. Wendy eats an omelet for lunch.
Question:
Vladimir must eat which one of the following foods?
Choices:
(A) fish
(B) hot cakes
(C) macaroni
(D) omelet
(E) poached eggs
###
# Declarations
people = EnumSort([Vladimir, Wendy])
meals = EnumSort([breakfast, lunch, dinner, snack])
foods = EnumSort([fish, hot_cakes, macaroni, omelet, poached_eggs])
eats = Function([people, meals] -> [foods])

# Constraints
ForAll([m:meals], eats(Vladimir, m) != eats(Wendy, m)) ::: At no meal does Vladimir
        eat the same kind of food as Wendy
ForAll([p:people, f:foods], Count([m:meals], eats(p, m) == f) <= 1) ::: Neither of
       them eats the same kind of food more than once during the day
ForAll([p:people], Or(eats(p, breakfast) == hot_cakes, eats(p, breakfast) ==
       poached_eggs, eats(p, breakfast) == omelet)) ::: For breakfast, each eats
       exactly one of the following: hot cakes, poached eggs, or omelet
ForAll([p:people], Or(eats(p, lunch) == fish, eats(p, lunch) == hot_cakes, eats(p,
       lunch) == macaroni, eats(p, lunch) == omelet)) ::: For lunch, each eats
       exactly one of the following: fish, hot cakes, macaroni, or omelet
ForAll([p:people], Or(eats(p, dinner) == fish, eats(p, dinner) == hot_cakes, eats(p
       , dinner) == macaroni, eats(p, dinner) == omelet)) ::: For dinner, each eats
       exactly one of the following: fish, hot cakes, macaroni, or omelet
ForAll([p:people], Or(eats(p, snack) == fish, eats(p, snack) == omelet)) ::: For a
       snack, each eats exactly one of the following: fish or omelet
eats(Wendy, lunch) == omelet ::: Wendy eats an omelet for lunch

# Options
Question ::: Vladimir must eat which one of the following foods?
is_valid(Exists([m:meals], eats(Vladimir, m) == fish)) ::: (A)
is_valid(Exists([m:meals], eats(Vladimir, m) == hot_cakes)) ::: (B)
is_valid(Exists([m:meals], eats(Vladimir, m) == macaroni)) ::: (C)
is_valid(Exists([m:meals], eats(Vladimir, m) == omelet)) ::: (D)
is_valid(Exists([m:meals], eats(Vladimir, m) == poached_eggs)) ::: (E)
------
Problem:
In a repair facility there are exactly six technicians: Stacy, Urma, Wim, Xena,
       Yolanda, and Zane. Each technician repairs machines of at least one of the
       following three types-radios, televisions, and VCRs-and no other types. The
       following conditions apply: Xena and exactly three other technicians repair
       radios. Yolanda repairs both televisions and VCRs. Stacy does not repair any
       type of machine that Yolanda repairs. Zane repairs more types of machines
       than Yolanda repairs. Wim does not repair any type of machine that Stacy
       repairs. Urma repairs exactly two types of machines.
Question:
Which one of the following pairs of technicians could repair all and only the same
       types of machines as each other?
Choices:
(A) Stacy and Urma
(B) Urma and Yolanda
(C) Urma and Xena
(D) Wim and Xena
(E) Xena and Yolanda
###
# Declarations
technicians = EnumSort([Stacy, Urma, Wim, Xena, Yolanda, Zane])
machines = EnumSort([radios, televisions, VCRs])
repairs = Function([technicians, machines] -> [bool])

# Constraints
ForAll([t:technicians], Count([m:machines], repairs(t, m)) >= 1) ::: each
       technician repairs machines of at least one of the following three types
And(repairs(Xena, radios), Count([t:technicians], And(t != Xena, repairs(t, radios)
       )) == 3) ::: Xena and exactly three other technicians repair radios
And(repairs(Yolanda, televisions), repairs(Yolanda, VCRs)) ::: Yolanda repairs both
        televisions and VCRs
ForAll([m:machines], Implies(repairs(Yolanda, m), Not(repairs(Stacy, m)))) :::
       Stacy does not repair any type of machine that Yolanda repairs
Count([m:machines], repairs(Zane, m)) > Count([m:machines], repairs(Yolanda, m))
       ::: Zane repairs more types of machines than Yolanda repairs
ForAll([m:machines], Implies(repairs(Stacy, m), Not(repairs(Wim, m)))) ::: Wim does
        not repair any type of machine that Stacy repairs
Count([m:machines], repairs(Urma, m)) == 2 ::: Urma repairs exactly two types of
       machines

# Options
Question ::: ::: Which one of the following pairs of technicians could repair all
       and only the same types of machines as each other?
is_sat(ForAll([m:machines], repairs(Stacy, m) == repairs(Urma, m))) ::: (A)
is_sat(ForAll([m:machines], repairs(Urma, m) == repairs(Yolanda, m))) ::: (B)
is_sat(ForAll([m:machines], repairs(Urma, m) == repairs(Xena, m))) ::: (C)
is_sat(ForAll([m:machines], repairs(Wim, m) == repairs(Xena, m))) ::: (D)
is_sat(ForAll([m:machines], repairs(Xena, m) == repairs(Yolanda, m))) ::: (E)
------
Problem:
Workers at a water treatment plant open eight valves-G, H, I, K, L, N, O, and P-to
       flush out a system of pipes that needs emergency repairs. To maximize safety
       and efficiency, each valve is opened exactly once, and no two valves are
       opened at the same time. The valves are opened in accordance with the
       following conditions: Both K and P are opened before H. O is opened before L
       but after H. L is opened after G. N is opened before H. I is opened after K.
Question: Each of the following could be the fifth valve opened EXCEPT:
Choices:
(A) H
(B) I
(C) K
```

```
(D) N
(E) O
"""
# Declarations
valves = EnumSort([G, H, I, K, L, N, O, P])
opened = Function([valves] -> [int])
ForAll([v:valves], And(1 <= opened(v), opened(v) <= 8))

# Constraints
Distinct([v:valves], opened(v)) ::: no two valves are opened at the same time
And(opened(K) < opened(H), opened(P) < opened(H)) ::: Both K and P are opened
      before H
And(opened(O) > opened(H), opened(O) < opened(L)) ::: O is opened before L but
      after H
opened(L) > opened(G) ::: L is opened after G
opened(N) < opened(H) ::: N is opened before H
opened(I) > opened(K) ::: I is opened after K

# Options
Question ::: Each of the following could be the fifth valve opened EXCEPT:
is_exception(is_sat(opened(H) == 5)) ::: (A)
is_exception(is_sat(opened(I) == 5)) ::: (B)
is_exception(is_sat(opened(K) == 5)) ::: (C)
is_exception(is_sat(opened(N) == 5)) ::: (D)
is_exception(is_sat(opened(O) == 5)) ::: (E)
------
Problem:
[[PROBLEM]]
Question:
[[QUESTION]]
Choices:
[[CHOICES]]
###
```

F. Hallucinated output of Llama-2-13b-hf for ProntoQA dataset

```
"id": "ProntoQA_15",
"context": "Jompuses are not small. Jompuses are tumpuses. Tumpuses are not
      kind. Each tumpus is a vumpus. Vumpuses are metallic. Vumpuses are
      numpuses. Each numpus is fruity. Each numpus is a dumpus. Dumpuses are
      nervous. Dumpuses are rompuses. Each rompus is opaque. Every wumpus is
      dull. Each rompus is a zumpus. Each zumpus is hot. Every zumpus is an
      impus. Every impus is not dull. Each impus is a yumpus. Sam is a numpus
      .",
"question": "Is the following statement true or false? Sam is not dull.",
"answer": "A",
"program": "import { Component, OnInit } from '@angular/core';\nimport { Router
       } from '@angular/router';\nimport { FormBuilder, FormGroup, Validators }
       from '@angular/forms';\nimport { AuthService } from '../auth.service';\
      nimport { ToastrService } from 'ngx-toastr';\n\n@Component({\n  selector:
       'app-login',\n  templateUrl: './login.component.html',\n  styleUrls:
      ['./login.component.css']\n})\nexport class LoginComponent implements
      OnInit {\n\n  loginForm: FormGroup;\n  loading = false;\n\n  constructor(
      private formBuilder: FormBuilder, private router: Router, private
      authService: AuthService, private toastr: ToastrService) { }\n\n
      ngOnInit() {\n    this.loginForm = this.formBuilder.group({\n      email:
       ['', [Validators.required, Validators.email]],\n      password: ['', [
      Validators.required]]\n    });\n  }\n\n  onSubmit() {\n    this.loading =
       true;\n    this.authService.login(this.loginForm.value).subscribe(\n
          data => {\n        this.router.navigate(['/dashboard']);\n
      this.toastr.success('Login Successful', 'Welcome to the Dashboard');\n
          },\n      error => {\n        this.toastr.error('Login Failed', '
      Please check your credentials');\n        this.loading = false;\n      }\
      n    );\n  }\n\n}",
"flag": "parsing error",
"predicted_answer": "A"
```

```
"id": "ProntoQA_16",
"context": "Yumpuses are hot. Each yumpus is a rompus. Rompuses are happy.
      Rompuses are impuses. Each impus is not amenable. Every impus is a dumpus
      . Dumpuses are opaque. Dumpuses are tumpuses. Numpuses are small.
      Tumpuses are orange. Every tumpus is a wumpus. Wumpuses are not small.
      Wumpuses are vumpuses. Every vumpus is fruity. Every vumpus is a jompus.
      Jompuses are not dull. Every jompus is a zumpus. Sally is a rompus.",
"question": "Is the following statement true or false? Sally is small.",
"answer": "B",
"program": "#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n#\n
      #\n#\n#\n#\n#\n#\n#\n#\n#\n#",
"flag": "parsing error",
"predicted_answer": "B",
```

G. Correct output of Llama-2-13b-hf for ProntoQA dataset

```
"id": "ProntoQA_14",
"context": "Jompuses are large. Every jompus is a zumpus. Each zumpus is sweet.
       Zumpuses are numpuses. Every numpus is hot. Each tumpus is opaque.
      Numpuses are yumpuses. Every yumpus is brown. Each yumpus is a wumpus.
      Wumpuses are not opaque. Wumpuses are impuses. Fae is a jompus.",
"question": "Is the following statement true or false? Fae is opaque.",
"answer": "B",
"program": "$}}%\nPredicates:\nJompus(x) ::: x is a jompus\nLarge(x) ::: x is
      large\nZumpus(x) ::: x is a zumpus\nSweet(x) ::: x is sweet\nNumpus(x)
      ::: x is a numpus\nHot(x) ::: x is hot\nTumpus(x) ::: x is a tumpus\
      nOpaque(x) ::: x is opaque\nYumpus(x) ::: x is a yumpus\nBrown(x) ::: x
      is brown\nWumpus(x) ::: x is a wumpus\nImpus(x) ::: x is an impus\nFae(x)
       ::: x is Fae\nPremises:\n∀x (Jompus(x) → Large(x)) ::: Jompuses are
      large.\n∀x (Jompus(x) → Zumpus(x)) ::: Every jompus is a zumpus.\n∀x (
      Zumpus(x) → Sweet(x)) ::: Each zumpus is sweet.\n∀x (Zumpus(x) → Numpus(x
      )) ::: Zumpuses are numpuses.\n∀x (Numpus(x) → Hot(x)) ::: Every numpus
      is hot.\n∀x (Tumpus(x) → Opaque(x)) ::: Each tumpus is opaque.\n∀x (
      Numpus(x) → Yumpus(x)) ::: Numpuses are yumpuses.\n∀x (Yumpus(x) → Brown(
      x)) ::: Every yumpus is brown.\n∀x (Yumpus(x) → Wumpus(x)) ::: Each
      yumpus is a wumpus.\n∀x (Wumpus(x) → ¬Opaque(x)) ::: Wumpuses are not
      opaque.\n∀x (Wumpus(x) → Impus(x)) ::: Wumpuses are impuses.\nJompus(fae)
       ::: Fae is a jompus.\nConclusion:\nOpaque(fae) ::: Fae is opaque.\n
      ------",
"flag": "success",
"predicted_answer": "B"
```