In the Mood for Inference: Logic-Based Natural Language Inference with Large Language Models

Bill Noble[†] and **Rasmus Blanck**[†] and **Gijs Wijnholds**[‡]
[†]Dept. of Philosophy, Linguistics and Theory of Science, University of Gothenburg
[‡]Leiden Institute of Advanced Computer Science, Leiden University
bill.noble@gu.se, rasmus.blanck@gu.se, g.j.wijnholds@liacs.leidenuniv.nl

Abstract

In this paper we explore a hybrid approach to challenging Natural Language Inference datasets that combines Large Language Models (LLMs) and logical theorem proving. We report on an experiment which combines an LLM meta-prompting strategy, eliciting logical representations, and Prover9, a first-order logic theorem prover. In addition, we experiment with the inclusion of (logical) world knowledge. Our findings suggest that (i) requesting first-order logic formalizations of sentences usually improves model performance, even when those formulas are not explicitly used, (ii) determining the inference relation from the generated formulas nevertheless performs worse, and (iii) priming the model to generate relative world knowledge is sometimes effective. We argue that these results explicate the weaknesses of both approaches. As such, we consider this study a source of inspiration for future work in the field of neuro-symbolic reasoning.

1 Introduction

Natural Language Inference (NLI) is a core task in Natural Language Processing (NLP) and is often presented as a proxy measure of the reasoning capabilities of NLP models. Briefly, a model is presented with a premise sentence and a hypothesis sentence and must decide whether the hypothesis is entailed by the premise (E), contradicts it (C), or is neutral with respect to it (N).

Although many NLI datasets have been developed, starting with the SICK dataset of Marelli et al. (2014) and the SNLI dataset of Bowman et al. (2015), more attention has recently been put on using NLI to measure specific linguistic phenomena. The MED dataset, for example, tests for monotonic reasoning (Yanaka et al., 2019a; Richardson et al., 2020). The CURRICULUM benchmark (Chen and Gao, 2022) is a notable aggregation of NLU tasks (including things like question answering) that have been uniformly formulated as NLI tasks.[1]

One often heard criticism of NLI as a task is that datasets often contain biases and annotation artifacts, and that models trained on them exhibit poor generalization capabilities. It is shown by Yanaka et al. (2019a) for English, and corroborated by Wijnholds (2023) for Dutch, that models have a tendency to overtune, and that they fail to properly address negation. That models seem to exploit relatively shallow heuristics such as lexical overlap and sentence length, is confirmed in prior work (Naik et al., 2018; McCoy et al., 2019), and an effort to repair an existing dataset is done by Kalouli et al. (2023). Finally, NLI models don't necessarily transfer gracefully to other NLI datasets (Talman and Chatzikyriakidis, 2019; Bhargava et al., 2021).

Many of the mentioned datasets and results were achieved with encoder-only models like BERT, which can narrowly generalize through finetuning; gradually, this has been replaced by decoder-only Large Language Models (GPT-3 onward), allowing for the NLI task to be stated as a text-to-text problem. Though this avoids some of the above-mentioned pitfalls, the results of McKenna et al. (2023) show that generative language models still suffer from bias and additionally are a source of *hallucinations*, an issue that is persistent for models that are effectively next-word predictors.

In order to control the output of model prompting, one method is to specifically *constrain* model output as a part of the decoding scheme; additionally this has the benefit of guaranteeing syntactic correctness over prompting results, leading to more

[1]A historical note: Prior to the advent of neural (language) models, Recognizing Textual Entailment (RTE) was the more common terminology for inference datasets. These datasets, such as the FraCaS suite (Cooper et al., 1996), typically framed the task as a two-way classification (entailment vs. non-entailment) with canonical examples for different linguistic phenomena, but there is no hard distinction between NLI and RTE. In the continuation, we use the NLI/RTE terminology primarily to distinguish between three- and two-label datasets.

effective prompting strategies. Constrained decoding approaches can work either through vocabulary filters (e.g. only 'E', 'N' and 'C' are valid prompt continuations), or through more sophisticated strategies like generating vocabulary filters determined by finite state automata (aka regular expressions) (Willard and Louf, 2023) or context-free grammars (Beurer-Kellner et al., 2024). While these approaches provide some control over model output, they are nevertheless limited to *syntactic* correctness, meaning they will not fully avoid hallucinations. In this work we use vocabulary filters.

Mixing LLM prompting with logic-based approaches is an emerging field with a number of precedents in NLP. A recent example is the study of Pan et al. (2023), which combines LLM prompting with theorem proving for logical reasoning, with the downside of returning incorrect representations back to the LLM to repeat the prompting procedure. Another work suggests constrained decoding for a variety of (structured) NLP tasks (Geng et al., 2023), but unfortunately doesn't provide a concrete implementation for most examples.

While there is some recent work attempting to bring logical representations in the loop in order to formalize the (chain-of-thought) prompting process (Ranaldi et al., 2025), logic-based approaches to NLI are rare, and were mostly performed in the era before LLMs, typically in a multimodal setting or following a pipeline where sentences are first encoded using a syntactic and semantic parser, after which a classification is made (Abzianidze, 2020; Abzianidze and Kogkalidis, 2021; Chen et al., 2021; Suzuki et al., 2019; Tomihari and Yanaka, 2023).

In this work we set out to provide a pilot study mixing the above approaches to tackle complex NLI test sets with a variety of strategies, including prompting LLMs for first-order logical representations. [2]

2 Logic-Based NLI with an LLM

Concretely, our pipeline works as follows: We prompt a model to generate logical representations for a given premise–hypothesis pair, after which we re-prompt the model to generate a (constrained) answer on the (non-)entailment between the premise and hypothesis. Whenever relevant, we feed the generated formulas to a theorem prover[3] to assess which path is more performant. As a baseline we consider a *label only* strategy where the model is not prompted for any logical representations but must directly generate the NLI label.

Datasets We evaluate our approach on six different sections of the CURRICULUM benchmark (Chen and Gao, 2022). The **comparative**, **conditional**, **negation**, and **quantifiers** datasets are drawn from Richardson et al. (2020) and follow the NLI format. The **lexical entailment** section has test set items drawn from Schmitt and Schütze (2021) and Glockner et al. (2018), and the **monotonicity** section has test set items drawn from Yanaka et al. (2019b) and Richardson et al. (2020). These later two sections use the RTE format.

Prompt setup Our prompting approach is an example of meta-prompting, i.e., the outcome of the first prompt is included in a second prompt. We distinguish three alternatives: Firstly, the *label only* prompt asks the model to directly generate an NLI label, constrained to the three possibilities (E, N, C). Second, the *formula* prompt asks the model to first generate formulas in first-order logic (in a model-friendly format) for the premise and hypothesis and then is asked to generate a label, hence incorporating both textual and logical representations of the NLI instance. Finally, there is the *formulas and world knowledge* setting where the intermediate generation prompt provides formulas and relevant world knowledge in logical form.

Model choice Given that we strive for full transparency in our experiments, we set four desiderata (in order of importance) and choose such that the model (1) has freely available architecture code; (2) has fully specified training data; (3) has a reasonable performance baseline; and (4) is as small as possible modulo the preceding points. Given these constraints, we work with Zephyr[4], which strikes a balance between performance and model size, and is fully transparent in terms of architecture code and training data.

3 Results

Table 1 displays the overall results for several NLI datasets. Additionally, results on the two RTE tasks are given in Table 2.

[2] The code is available online: https://github.com/GU-CLASP/logic-based-NLI-with-LLMs/

[3] Prover9, (McCune, 2005–2010)

[4] https://huggingface.co/HuggingFaceH4/zephyr-7b-beta

	Label	**E**		**C**		**N**	
		DA	P9	DA	P9	DA	P9
Dataset	*Prompt*						
comparative	label-only	77.6	–	73.7	–	6.8	–
	forms	71.4	8.2	**94.9**	0.0	**17.5**	99.0
	forms+wk	**98.0**	100.0	56.6	0.0	9.7	0.0
conditional	label-only	48.4	–	0.0	–	50.9	–
	forms	**74.7**	59.8	**54.7**	50.0	**43.6**	100.0
	forms+wk	50.5	37.0	51.6	48.4	0.9	100.0
negation	label-only	0.0	–	15.6	–	36.5	–
	forms	0.0	0.0	60.0	98.9	**90.4**	100.0
	forms+wk	**2.2**	0.0	**62.2**	98.8	9.6	98.2
quantifier	label-only	70.0	–	3.5	–	**96.9**	–
	forms	72.2	59.8	**82.5**	27.2	29.2	100.0
	forms+wk	**98.9**	64.0	1.8	23.5	5.2	100.0

Table 1: Mean accuracy (recall) by label for Zephyr on NLI datasets, using different prompt schemes. Where relevant, accuracy is shown for both the LLM's direct answer (**DA**) and the label inferred from the generated formulas and the Prover9 theorem prover (**P9**). For each dataset and label, the best DA result is **bolded** and the P9 result in underlined when it exceeds the DA result. The P9 column excludes items that resulted in a Prover9 error (see Table 3 for the unfiltered results).

	Label	**E**		**N/C**	
		DA	P9	DA	P9
Dataset	*Prompt*				
lexical	label-only	6.7	–	94.6	–
	forms	**25.0**	1.5	**97.3**	100.0
	forms+wk	**25.0**	65.3	96.6	34.8
monotonicity	label-only	**58.3**	–	34.8	–
	forms	46.8	16.7	**47.8**	90.9
	forms+wk	47.5	50.0	47.2	54.6

Table 2: As Table 1 but for the RTE datasets. See Table 4 for the unfiltered Prover9 results.

In the mood for logic We firstly note that the *direct answer* performance is generally higher in the setup in which the model is asked to generate a logic formula (*formulas*) and subsequentially generate the NLI label. In other words: When the model is *self-primed* to think in terms of logic, it appears to label in terms of logic, generally improving performance. A notable exception to this rule is the case of neutral-labeled items in the **quantifier** dataset, where the *label-only* setup performed significantly better (96.9%) than either of the two prompt schemes involving formulas (29.9% for *formulas* and 5.2 for *formulas + world knowledge*). For this dataset, it seems that generating formulas causes the model to predict a logical relationship between sentences where in fact none exists.

Priming the model to generate formulas for relevant world knowledge helps in some cases, but the effect is much more mixed. For example, it is beneficial for entailments in the **comparative** and **quantifier** datasets, but detrimental for the other two labels in the same datasets. It is possible that in these cases, asking the model to generate world knowledge biases it more towards finding an entailment in general.

Theorem proving We secondly note that the labels inferred from the generated formulas are not always an improvement over the model's direct answer. For the neutral (or non-entailment) columns, we see that the Prover9-inferred label accuracy is typically higher than the direct answer (often much higher), but recall that we infer a neutral label whenever Prover9 cannot find a proof of the hypothesis (or its negation) from the premise and any relevant world-knowledge formulas, so these apparently good results for neutral items may just be evidence that the generated formulas don't fully capture the logical relationships that would be required to draw an inference if there were one. This can happen when the model produces formulas that are unrelated to each other for the wrong reasons (e.g., inconsistently translated predicates).

There are, however, several other cases where the inferred label accuracy shows a notable performance improvement over the direct answer. Prover9 accuracy is significantly better for contradiction-labeled items in the **negation** dataset, and it is somewhat better for entailments of the **lexical** dataset in the prompt scheme that includes world-knowledge formulas.

Overall, while there are some cases in which the label inferred from the generated formulas outperforms the direct-answer label, there are even more cases where generating the formulas improves the direct answer but the formulas themselves cannot be used to infer the correct label. This suggests that the utility of prompting the model for formulas is mostly in priming it to attend to the logical relationships between the natural language sentences. Upon closer inspection, we suspect that the logical representations for the premise and hypothesis are often not linked together logically, pushing the theorem prover towards Neutral.

4 Conclusion

This work investigates the use of constrained decoding and LLM prompting for Natural Language Inference. We specifically test three setups: (1) An LLM is prompted to solve the task directly; (2) the LLM first is prompted to generate logic formulas and subsequently re-prompted to use those formulas to provide an answer; and (3) the model is also prompted to generate formulas capturing any relevant lexical knowledge before answering. Generated formulas are also fed to a theorem prover. We observe that while the theorem prover may help in cases of entailment and non-entailment, the overall performance is highest for the two-step prompting approach of letting the model decide based on its own generated formulas.

Limitations We consider this work a pilot study investigating the applicability of constrained decoding to support NLI systems based on LLM prompting. This leads to a number of lessons of this work: (1) Priming an LLM by asking for logical representations increases performance on challenging NLI test sets; (2) Generating logical representations with only prompt examples as gold standard is too primitive to use in combination with theorem proving; (3) Adding world knowledge can mitigate the gold standard issue; (4) With the adequate combination of representation format, language model, and prompt setup one may push the limits of NLI; (5) Constrained decoding can play a role in controlling LLM output and overall performance.

Future Work The findings in this paper warrant a lot of future work; for example, the lack of gold standard data in the test sets we used makes it difficult for the model to tune its generated logical representations, so ideally a gold standard is required.

Another underrepresented concept is a change in representation format, where predicate logic formulas could be encoded in formats more represented in LLM training data, such as Python code, or Z3 statements. Natural logic may also be a promising output format for LLMs due to its closeness to natural language (Lakoff, 1970).

Acknowledgments

This work was supported by grant 2014-39 from the Swedish Research Council (VR) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Lasha Abzianidze. 2020. Learning as abduction: Trainable natural logic theorem prover for natural language inference. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 20–31, Barcelona, Spain (Online). Association for Computational Linguistics.

Lasha Abzianidze and Konstantinos Kogkalidis. 2021. A logic-based framework for natural language inference in dutch. *arXiv preprint arXiv:2110.03323*.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Forty-first International Conference on Machine Learning*.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in NLI: Ways (not) to go beyond simple heuristics. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 125–135, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Zeming Chen and Qiyue Gao. 2022. Curriculum: A broad-coverage benchmark for linguistic phenomena in natural language understanding. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3204–3219, Seattle, United States. Association for Computational Linguistics.

Zeming Chen, Qiyue Gao, and Lawrence S. Moss. 2021. NeuralLog: Natural language inference with joint neural and logical reasoning. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 78–88, Online. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Aikaterini-Lida Kalouli, Hai Hu, Alexander F. Webb, Lawrence S. Moss, and Valeria de Paiva. 2023. Curing the SICK and other NLI maladies. *Computational Linguistics*, 49(1):199–243.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1):151–271.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

W. McCune. 2005–2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Hosseini, Mark Johnson, and Mark Steedman. 2023. Sources of hallucination by large language models on inference tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2758–2774, Singapore. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353,

Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Leonardo Ranaldi, Marco Valentino, Alexander Polonsky, and Andrè Freitas. 2025. Improving chain-of-thought reasoning via quasi-symbolic abstractions. *arXiv preprint arXiv:2502.12616*.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.

Martin Schmitt and Hinrich Schütze. 2021. Language Models for Lexical Inference in Context. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1267–1280, Online. Association for Computational Linguistics.

Riko Suzuki, Hitomi Yanaka, Masashi Yoshikawa, Koji Mineshima, and Daisuke Bekki. 2019. Multimodal logical inference system for visual-textual entailment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 386–392, Florence, Italy. Association for Computational Linguistics.

Aarne Talman and Stergios Chatzikyriakidis. 2019. Testing the generalization power of neural network models across NLI benchmarks. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 85–94, Florence, Italy. Association for Computational Linguistics.

Akiyoshi Tomihari and Hitomi Yanaka. 2023. Logic-based inference with phrase abduction using vision-and-language models. *IEEE Access*, 11:45645–45656.

Gijs Wijnholds. 2023. Assessing monotonicity reasoning in Dutch through natural language inference. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1494–1500, Dubrovnik, Croatia. Association for Computational Linguistics.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv e-prints*, pages arXiv–2307.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019a. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019b. Can Neural Networks Understand Monotonicity Reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

A LLM prompts

Figures 1, 2, and 3 show the three different prompt templates used to elicit NLI classifications from the LLM. Only the `ITEM_PREMISE` and `ITEM_HYPOTHESIS` vary by item, whereas the `DATASET_LABELS` and `EXAMPLE_*` fields vary by dataset. The `LM_*` fields are filled in by the LLM.

The specific few-shot examples used for each dataset can be found in Figures 4–9.

```
Given a pair of sentences, the task is to determine whether Sentence A entails Sentence B by labeling
    ↪  the pair with <DATASET_LABELS>.

Sentence A: <EXAMPLE_1_PREMISE>
Sentence B: <EXAMPLE_1_HYPOTHESIS>
###
The relation between Sentence A and Sentence B is: <EXAMPLE_1_LABEL>
∼ REPEATED FOR EXAMPLES 2 and 3 ∼

Sentence A: <ITEM_PREMISE>
Sentence B: <ITEM_HYPOTHESIS>
###
The relation between Sentence A and Sentence B is: <LM_DIRECT_ANSWER>
```

Figure 1: The prompt asking the model for a direct answer. <DATASET_LABELS> is adaptable, depending on whether the task at hand uses binary (RTE) or ternary (NLI) classification; <LM_DIRECT_ANSWER> is constrained by the relevant label set; the three few-shot examples are specific to the dataset the item comes from (see Figures 4-9).

```
 Given a pair of sentences, the task is to parse each sentence into first-order logic formulas and
     ↪ then determine whether Sentence A entails Sentence B by labeling the pair with
     ↪ <DATASET_LABELS>.
The grammar of first-order logic formulas is defined as follows:
1) logical conjunction of expr1 and expr2: expr1 & expr2
2) logical disjunction of expr1 and expr2: expr1 | expr2
3) logical negation of expr1: -expr1
5) expr1 implies expr2: expr1 -> expr2
6) expr1 if and only if expr2: expr1 <-> expr2
7) logical universal quantification over expr1: forall x. expr1
8) logical existential quantification over expr1: exists x. expr1

Sentence A: <EXAMPLE_1_PREMISE>
Sentence B: <EXAMPLE_1_HYPOTHESIS>
###
Formula A: <EXAMPLE_1_PREMISE> ::: <EXAMPLE_1_PREMISE_FORMULA>
Formula B: <EXAMPLE_1_PREMISE> ::: <EXAMPLE_1_HYPOTHESIS_FORMULA>
###
The relation between Sentence A and Sentence B is: <EXAMPLE_1_LABEL>
∼ REPEATED FOR EXAMPLES 2 and 3 ∼

Sentence A: <ITEM_PREMISE>
Sentence B: <ITEM_HYPOTHESIS>
###
Formula A: <ITEM_PREMISE> ::: <LM_PREMISE_FORMULA>
Formula B: <ITEM_HYPOTHESIS> ::: <LM_HYPOTHESIS_FORMULA>
###
The relation between Sentence A and Sentence B is: <LM_DIRECT_ANSWER>
```

Figure 2: The prompt asking the model for first-order logic formulas for the premise and hypothesis, as well as a direct answer. <DATASET_LABELS> is adaptable, depending on whether the task at hand uses binary (RTE) or ternary (NLI) classification; <LM_DIRECT_ANSWER> is constrained by the relevant label set; the three few-shot examples are specific to the dataset the item comes from (see Figures 4-9).

```
Given a pair of sentences, the task is to parse each sentence into first-order logic formulas, then
    ↪ write first-order logic formulas that capture any relevant lexical knowledge, and finally
    ↪ determine whether Sentence A entails Sentence B by labeling the pair with <DATASET_LABELS>.
1) logical conjunction of expr1 and expr2: expr1 & expr2
2) logical disjunction of expr1 and expr2: expr1 | expr2
3) logical negation of expr1: -expr1
5) expr1 implies expr2: expr1 -> expr2
6) expr1 if and only if expr2: expr1 <-> expr2
7) logical universal quantification over expr1: forall x. expr1
8) logical existential quantification over expr1: exists x. expr1

Sentence A: <EXAMPLE_1_PREMISE>
Sentence B: <EXAMPLE_1_HYPOTHESIS>
###
Formula A: <EXAMPLE_1_PREMISE> ::: <EXAMPLE_1_PREMISE_FORMULA>
Formula B: <EXAMPLE_1_PREMISE> ::: <EXAMPLE_1_HYPOTHESIS_FORMULA>
###
Lexical knowledge:
<EXAMPLE_1_WORLD_KNOWLEDGE_FORMULA_1>
. . .
<EXAMPLE_1_WORLD_KNOWLEDGE_FORMULA_N>
###
The relation between Sentence A and Sentence B is: <EXAMPLE_1_LABEL>
~ REPEATED FOR EXAMPLES 2 and 3 ~

Sentence A: <ITEM_PREMISE>
Sentence B: <ITEM_HYPOTHESIS>
###
Formula A: <ITEM_PREMISE> ::: <LM_PREMISE_FORMULA>
Formula B: <ITEM_HYPOTHESIS> ::: <LM_HYPOTHESIS_FORMULA>
###
Lexical knowledge:
<LM_WORLD_KNOWLEDGE_FORMULAS>
###
The relation between Sentence A and Sentence B is: <LM_DIRECT_ANSWER>
```

Figure 3: The prompt asking the model for first-order logic formulas for the premise and hypothesis, as well as a direct answer. <DATASET_LABELS> is adaptable, depending on whether the task at hand uses binary (RTE) or ternary (NLI) classification; <LM_DIRECT_ANSWER> is constrained by the relevant label set; <LM_WORLD_KNOWLEDGE_FORMULAS> is constrained to be a newline-separated list of strings; the three few-shot examples are specific to the dataset the item comes from (see Figures 4-9).

```
EXAMPLE_1 = { # example id 4598
    'PREMISE' = "The purple alien drank soda."
    'HYPOTHESIS' = "The purple alien drank coke."
    'PREMISE_FORMULA' = "exists x. exists y. Purple(x) & Alien(x) & Soda(y) & Drank(x, y)"
    'HYPOTHESIS_FORMULA' = "exists x. exists y. Purple(x) & Alien(x) & Coke(y) & Drank(x, y)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. (Coke(x) -> Soda(x)),
    ]
}

EXAMPLE_2 = { # example id 5075
    'PREMISE' = "Nobody danced."
    'HYPOTHESIS' = "Nobody moved."
    'PREMISE_FORMULA' = "forall x. -Dance(x)"
    'HYPOTHESIS_FORMULA' = "forall x. -Move(x)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. (Dance(x) -> Move(x)),
    ]
}

EXAMPLE_3 = { # example id 54
    'PREMISE' = "All animals like to scratch their ears."
    'HYPOTHESIS' = "All dogs like to scratch their ears."
    'PREMISE_FORMULA' = "forall x. (Animal(x) -> LikesToScratchEars(x, x))"
    'HYPOTHESIS_FORMULA' = "forall x. (Dog(x) -> LikesToScratchEars(x, x))"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. (Dog(x) -> Animal(x)),
    ]
}
```

Figure 4: Few-shot examples for items from the **monotonicity** dataset. CURRICULUM item ids: 4598 5075 54

```
EXAMPLE_1 = { # example id 2675
    'PREMISE' = "Ruben is as tall as Jack , Jack is as tall as Francis , Francis is as tall as Gordon
        ↪  , Gordon is as tall as Bruce , Bruce is as tall as Alan , Alan is as tall as Danny ,
        ↪ Danny is taller than Allen"
    'HYPOTHESIS' = "Keith is taller than Alan"
    'PREMISE_FORMULA' = "AsTallAs(ruben, jack) & AsTallAs(jack, francis) & AsTallAs(francis, gordon)
        ↪ & AsTallAs(gordon, bruce) & AsTallAs(bruce, alan) & AsTallAs(alan, danny) & TallerThan(
        ↪ Danny, alan)"
    'HYPOTHESIS_FORMULA' = "TallerThan(keith, alan)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. forall y. TallerThan(x, y) -> -AsTallAs(y, x),
       forall x. forall y. forall z. (TallerThan(x, y) & TallerThan(y, z)) -> TallerThan(x, z),
    ]
}

EXAMPLE_2 = { # example id 648
    'PREMISE' = "Russell is taller than Oscar, Terrance, Lawrence, Dan, Felix, Todd, Alex, Jose and
        ↪ Harry , Russell is as tall as Clifton"
    'HYPOTHESIS' = "Felix is taller than Clifton"
    'PREMISE_FORMULA' = "TallerThan(russell, oscar) & TallerThan(russell, terrance) & TallerThan(
        ↪ russell, dan) & TallerThan(russell, felix) & TallerThan(russell, todd) & TallerThan(
        ↪ russell, alex) & TallerThan(russell, jose) & TallerThan(russell, harry) & AsTallAs(
        ↪ russell, clifton)"
    'HYPOTHESIS_FORMULA' = "TallerThan(felix, clifton)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. forall y. forall z. (AsTallAs(x, y) & TallerThan(x, z) -> TallerThan(y, z)),
       forall x. forall y. (TallerThan(x, y) -> -TallerThan(y, x)),
    ]
}

EXAMPLE_3 = { # example id 1421
    'PREMISE' = "Jesse is as tall as Paul , Paul is as tall as Terry , Terry is as tall as Sidney ,
        ↪ Sidney is as tall as Luis , Luis is as tall as Andy , Andy is as tall as Freddie ,
        ↪ Freddie is as tall as Adrian , Adrian is taller than James"
    'HYPOTHESIS' = "Luis is taller than James"
    'PREMISE_FORMULA' = "AsTallAs(jesse, paul) & AsTallAs(paul, terry) & AsTallAs(terry, sidney) &
        ↪ AsTallAs(sidney, luis) & AsTallAs(luis, andy) & AsTallAs(andy, freddie) & AsTallAs(
        ↪ freddie, adrian) & TallerThan(adrian, james)"
    'HYPOTHESIS_FORMULA' = "TallerThan(luis, james)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. forall y. forall z. (AsTallAs(x, y) & TallerThan(y, z) -> TallerThan(x, z)),
       forall x. forall y. forall z. (AsTallAs(x, y) & AsTallAs(y, z) -> AsTallAs(x, z)),
    ]
}
```

Figure 5: Few-shot examples for items from the **comparative** dataset. CURRICULUM item ids: 2675 648 1421

```
EXAMPLE_1 = { # example id 2200
    'PREMISE' = "Tony has not visited Beaverton, Johnny has not visited Long Beach, Ken has visited
        ↪ Kingston and if Tony has not visited Beaverton then Fred has not visited Danville"
    'HYPOTHESIS' = "Fred has not visited Danville"
    'PREMISE_FORMULA' = "-Visited(tony, beaverton) & -Visited(johnny, long_beach) & Visited(ken,
        ↪ kingston) & (-Visited(tony, beaverton) -> -Visited(fred, danville))"
    'HYPOTHESIS_FORMULA' = "-Visited(fred, danville)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}

EXAMPLE_2 = { # example id 2699
    'PREMISE' = "Felix has not visited Pampa, William has not visited Bessemer, Eddie has visited
        ↪ Grants Pass and if Felix has visited Pampa then Danny has visited Belmont"
    'HYPOTHESIS' = "Danny has visited Belmont"
    'PREMISE_FORMULA' = "-Visited(felix, pampa) & -Visited(william, bessemer) & Visited(eddie,
        ↪ grants_pass) & (Visited(felix, pampa) -> Visited(danny, belmont))"
    'HYPOTHESIS_FORMULA' = "Visited(danny, belmont)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}

EXAMPLE_3 = { # example id 1384
    'PREMISE' = "Don has not visited Norwich, Alberto has visited Nevada, Wallace has visited Wyoming
        ↪  and if Alberto has visited Nevada then Sam has visited Arcadia"
    'HYPOTHESIS' = "Sam has not visited Arcadia"
    'PREMISE_FORMULA' = "-Visited(don, norwhich) & Visited(alberto, nevada) & Visited(wallace,
        ↪ wyoming) & (Visited(alberto, nevada) -> Visited(sam, arcadia))"
    'HYPOTHESIS_FORMULA' = "-Visited(sam, arcadia)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}
```

Figure 6: Few-shot examples for items from the **conditional** dataset. CURRICULUM item ids: 2200 2699 1384

```
EXAMPLE_1 = { # example id 1933
    'PREMISE' = "A stricken butterfly has Wings."
    'HYPOTHESIS' = "A stricken butterfly wavers on Wings."
    'PREMISE_FORMULA' = "exists x. exists y.(Stricken(x) & Butterfly(x) & Wings(y) & Has(x, y))"
    'HYPOTHESIS_FORMULA' = "exists x. exists y.(Stricken(x) & Butterfly(x) & Wings(y) & WaversOn(x, y
        ↪ ))"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       forall x. forall y. (Has(x, y) -> WaversOn(x, y)),
    ]
}

EXAMPLE_2 = { # example id 3662
    'PREMISE' = "Sadat beat Jimmy Carter."
    'HYPOTHESIS' = "Jimmy Carter secluded Sadat."
    'PREMISE_FORMULA' = "Beat(sadat, jimmy_carter)"
    'HYPOTHESIS_FORMULA' = "Secluded(jimmy_carter, sadat)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       -(forall x. forall y. (Beat(x, y) -> Secluded(y, x))),
    ]
}
```

Figure 7: Few-shot examples for items from the **lexical** dataset. CURRICULUM item ids: 1933 3662

```
EXAMPLE_1 = { # example id 1874
    'PREMISE' = "Harry has only visited Bhutan, Curtis has only visited Ecuador, Roland has only
        ↪ visited Philippines, Tom has only visited Uganda, Darren has only visited Jordan, Byron
        ↪ has only visited Cameroon, Willie has only visited Vanuatu, Brett has only visited North
        ↪ Korea"
    'HYPOTHESIS' = "Curtis didn't visit Belize"
    'PREMISE_FORMULA' = "Visit(harry, bhutan) & forall x. (Visit(harry, x) -> x = bhutan) & Visit(
        ↪ curtis, ecuador) & forall x. (Visit(curtis, x) -> x = ecuador) & Visit(roland,
        ↪ philippines) & forall x. (Visit(roland, x) -> x = philippines) & Visit(tom, uganda) &
        ↪ forall x. (Visit(tom, x) -> x = uganda) & Visit(darren, jordan) & forall x. (Visit(darren,
        ↪  x) -> x = jordan) & Visit(byron, cameroon) & forall x. (Visit(byron, x) -> x = cameroon)
        ↪  & Visit(willie, vanuatu) & forall x. (Visit(willie, x) -> x = vanuatu) & Visit(brett,
        ↪ north_korea) & forall x. (Visit(brett, x) -> x = north_korea)"
    'HYPOTHESIS_FORMULA' = "-Visit(curtis, belize)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       belize != ecuador,
    ]
}

EXAMPLE_2 = { # example id 2526
    'PREMISE' = "Howard has only visited Croatia, Karl has only visited Kosovo"
    'HYPOTHESIS' = "Ross didn't visit Croatia"
    'PREMISE_FORMULA' = "Visit(howard, croatia) & forall x. (Visit(howard, x) -> x = croatia) & Visit
        ↪ (karl, kosovo) & forall x. (Visit(karl, x) -> x = kosovo)"
    'HYPOTHESIS_FORMULA' = "-Visit(ross, croatia)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
       howard != ross,
    ]
}

EXAMPLE_3 = { # example id 581
    'PREMISE' = "Thomas has only visited Romania, Adrian has only visited Tuvalu, Everett has only
        ↪ visited Djibouti, Marc has only visited Dominica, Don has only visited China, Nicholas
        ↪ has only visited Turkmenistan, Lonnie has only visited Iraq, Theodore has only visited
        ↪ North Korea, Andrew has only visited Nepal, Ken has only visited Saint Lucia, Terrence
        ↪ has only visited Liberia"
    'HYPOTHESIS' = "Ken didn't visit Saint Lucia"
    'PREMISE_FORMULA' = "Visit(thomas, romania) & forall x. (Visit(thomas, x) -> x = romania) & Visit
        ↪ (adrian, tuvalu) & forall x. (Visit(adrian, x) -> x = tuvalu) & Visit(everett, djibouti)
        ↪ & forall x. (Visit(everett, x) -> x = djibouti) & Visit(marc, dominica) & forall x. (
        ↪ Visit(marc, x) -> x = dominica) & Visit(don, china) & forall x. (Visit(don, x) -> x =
        ↪ china) & Visit(nicholas, turkmenistan) & forall x. (Visit(nicholas, x) -> x =
        ↪ turkmenistan) & Visit(lonnie, iraq) & forall x. (Visit(lonnie, x) -> x = iraq) & Visit(
        ↪ theodore, korea) & forall x. (Visit(theodore, x) -> x = korea) & Visit(andrew, nepal) &
        ↪ forall x. (Visit(andrew, x) -> x = nepal) & Visit(ken, saint_lucia) & forall x. (Visit(
        ↪ ken, x) -> x = saint_lucia) & Visit(terrence, liberia) & forall x. (Visit(terrence, x) ->
        ↪  x = liberia)"
    'HYPOTHESIS_FORMULA' = "-Visit(ken, saint_lucia)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}
```

Figure 8: Few-shot examples for items from the **negation** dataset. CURRICULUM item ids: 1874 2526 581

```
EXAMPLE_1 = { # example id 2857
    'PREMISE' = "Everyone has visited Lesotho, Botswana, Cambodia, Kyrgyzstan, Lithuania, Tonga,
        ↪ Suriname, Costa Rica, Thailand, Bangladesh, New Zealand, Nigeria, Pakistan, Palau, Libya,
        ↪  Bosnia & Herzegovinia, United Arab Emirates, Chad, Solomon Islands and Ireland"
    'HYPOTHESIS' = "That person there did visit Libya"
    'PREMISE_FORMULA' = "forall x. Visit(x,lesotho) & Visit(x, botswana) & Visit(x, cambodia) & Visit
        ↪ (x, kyrgyzstan) & Visit(x, lithuania) & Visit(x, tonga) & Visit(x, suriname) & Visit(x,
        ↪ costa_rica) & Visit(x, thailand) & Visit(x, bangladesh) & Visit(x, new_zealand) & Visit(x,
        ↪  nigeria) & Visit(x, pakistan) & Visit(x, palau) & Visit(x, libya) & Visit(x,
        ↪ bosnia_herzegovinia) & Visit(x, united_arab_emirates) & Visit(x, chad) & Visit(x,
        ↪ solomon_islands) & Visit(x, ireland)"
    'HYPOTHESIS_FORMULA' = "exists x. Visit(x, libya)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}

EXAMPLE_2 = { # example id 1121
    'PREMISE' = "Everyone has visited Togo, Saudi Arabia, Malta, Bosnia & Herzegovinia, Gabon, Sierra
        ↪  Leone, El Salvador, The Bahamas, Mongolia, Mali and Djibouti"
    'HYPOTHESIS' = "Roland didn't visit Gabon"
    'PREMISE_FORMULA' = "forall x. Visit(x, togo) & Visit(x, saudi_arabia) & Visit(x, malta) & Visit(
        ↪ x, bosnia_herzegovinia) & Visit(x, gabon) & Visit(x, sierra_leone) & Visit(x, el_salvador)
        ↪  & Visit(x, bahamas) & Visit(x, mongolia) & Visit(x, mali) & Visit(x, djibouti)"
    'HYPOTHESIS_FORMULA' = "-Visit(roland, gabon)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}

EXAMPLE_3 = { # example id 2850
    'PREMISE' = "Someone has visited every person and every place"
    'HYPOTHESIS' = "That person there didn't visit United States"
    'PREMISE_FORMULA' = "exists x. forall y. (Person(y) | Place(y)) -> Visit(x, y)"
    'HYPOTHESIS_FORMULA' = "exists x. Person(x) & -Visit(x, united_states)"
    'WORLD_KNOWLEDGE_FORMULAS' = [
    ]
}
```

Figure 9: Few-shot examples for items from the **quantifier** dataset. CURRICULUM item ids: 2857 1121 2850

B Prover9 Errors

The LLM generated largely syntactically correct formulas of first-order logic, especially when provided with few-shot examples. However, there were cases where the generated formulas resulted in an error when fed into Prover9. In Tables 1 and 2 the P9 columns present label-wise accuracy results that have been filtered to exclude items where Prover9 generated an error (i.e., the denominator of the accuracy metric does not include those items). In most cases, the difference is small—note that considering the filtered or un-filtered version never changes whether the Prover9 result is higher than the direct answer). For completeness, this section shows a comparison of the filtered and un-filtered results.

A total of 310 errors were encountered over 3 600 generated formula sets (300 for each of 6 datasets and 2 prompt schemes). The breakdown of errors encountered can be found in Table 5.

	Label	E		C		N	
		⛛	▽	⛛	▽	⛛	▽
Dataset	*Prompt*						
comparative	forms	8.2	8.2	0.0	0.0	99.0	99.0
	forms+wk	100.0	100.0	0.0	0.0	0.0	0.0
conditional	forms	57.9	59.8	48.4	50.0	100.0	100.0
	forms+wk	35.8	37.0	47.4	48.4	100.0	100.0
negation	forms	0.0	0.0	97.8	98.9	96.5	100.0
	forms+wk	0.0	0.0	93.3	98.8	95.7	98.2
quantifier	forms	57.8	59.8	24.6	27.2	97.9	100.0
	forms+wk	61.1	64.0	21.1	23.5	96.9	100.0

Table 3: Unfiltered (⛛) and filtered (▽) mean label-wise accuracy for NLI classification when using the LLM-generated formulas to infer the label with Prover9.

	Label	E		N/C	
		⛛	▽	⛛	▽
Dataset	*Prompt*				
lexical	forms	1.3	1.5	89.9	100.0
	forms+wk	52.0	65.3	27.2	34.8
monotonicity	forms	13.7	16.7	74.5	90.9
	forms+wk	36.0	50.0	36.6	54.6

Table 4: Unfiltered (⛛) and filtered (▽) mean label-wise accuracy for RTE classification when using the LLM-generated formulas to infer the label with Prover9.

Prover9 Error	forms	forms+wk
parsing error (unexpected symbol)	79	92
symbol used with multiple arities	25	65
symbol used as both relation and function	14	35

Table 5: Counts of error types encountered by Prover9 when given the LLM-generated formulas. A total of 1 800 sets of formulas were generated for each prompt scheme (*forms* and *forms+wk*).