Chain-of-Context Provides Common-Sense for Reasoning in Large Language Models

Shayan Pardis
MIT EECS
shayanp@mit.edu

A. Anas Chentouf
MIT Math & EECS
chentouf@mit.edu

Juan F. Atehortúa Paredes
MIT Math & EECS
ate@mit.edu

Abstract

The ability to draw logical conclusions from given premises, a process known as Logical Reasoning, is pivotal for advancing artificial general intelligence. Despite the remarkable success of Large Language Models (LLMs) in various tasks, they often fall short in complex reasoning scenarios. Addressing this gap, the LINC (Olausson and et al., 2023) introduces a systematic approach to incorporate formalized logical reasoning through automatic proof checkers. While LINC outperforms prior LLM-based methods, particularly in tasks involving chain-of-thought (CoT) reasoning, it exhibits vulnerability to specific failure modes. Specifically, certain reasoning tasks necessitate additional context akin to "common sense," implicit in human understanding but absent in text. This study builds upon the foundation laid by LINC, aiming to generate common sense premises for the proof checker, thus addressing the limitations associated with implicit context in logical reasoning tasks. Our contribution is to utilize CoT reasoning to generate and improve common sense premises using few-shot training at test time, which allows us to obtain impressive 95% accuracy with limited computational resources, suggesting that this technique holds potential.

1 Introduction

While achieving remarkable feats in language understanding and predictive abilities, most NLP models operate as 'black boxes' with no guarantee that their logic is correct - they lack the ability to formalize and compartmentalize logical thinking. This makes their arguments susceptible to logical fallacies and limits their ability to explain their reasoning - simply because they have no notion of 'formal logic'.

In order to develop NLP models capable of logical reasoning, a sound framework is needed to guarantee the correctness of the reasoning. A recent work at MIT, LINC (Olausson and et al., 2023), proposes one such framework: converting natural language (NL) to first-order logic (FOL) via a large language model, and then using theorem-proving techniques to verify the generated FOL. Although this framework achieves astounding results, there is a major caveat: the translated FOL sometimes misses the context that is common sense to humans.

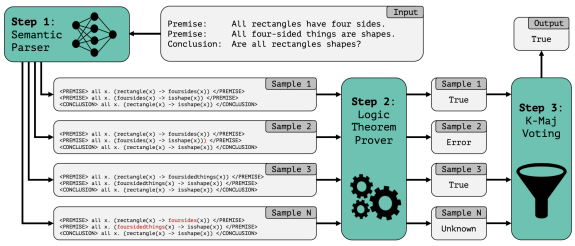


Figure 1: A general pipeline of the LINC approach, which takes an input of premises and conclusion, and uses a large language mode (LLM) in Step 1 as a semantic parser to convert the input into first-order logic in a few different ways. Each of the samples is then passed through a theorem prover, and the majority is output.

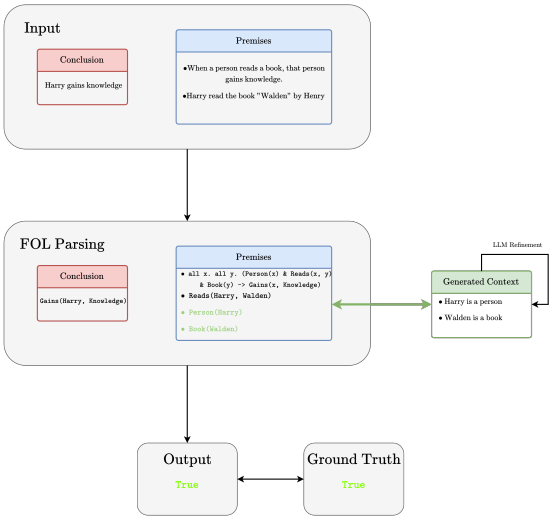


Figure 2: Our proposed approach: Add commonsense context and refine the result using LLM, which we can then integrate into LINC's pipeline. Refinement process makes sure that the translated FOL is valid. This can take multiple steps in which we pass the feedback of the theorem prover to the LLM to refine the result.

2 Methods

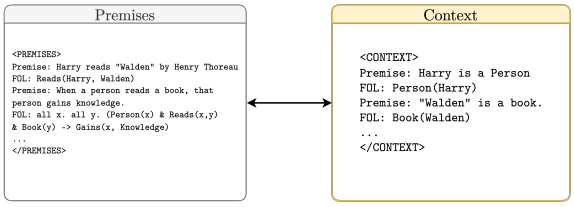
We define common sense knowledge as prior knowledge belief that is not necessarily explicitly stated in text, but implicitly understood by any reader. Common Ground (Stalnaker, 2002) elaborates on this concept. For instance, the sentence "Sam does not regret voting for Nader" presupposes that Sam voted for Nader. One of the biggest points of failure in LINC is precisely the lack of common sense. For example, "If something is cold, then it is not hot" and "Harry is a person" are both premises which may be crucial for a theorem prover to output a correct evaluation, but are never transcribed in the pipeline when premises such as "Harry reads a book" and "The coffee is cold" are transcribed to FOL.

2.1 Providing Failure mode examples for LLM

We try to leverage the human-like reasoning capabilities of LLMs to generate these additional missing premises (context) as an additional step before running the prover. As an attempt to remedy the common failure modes in LINC, we prompt the LLM with the following examples of common sense context.

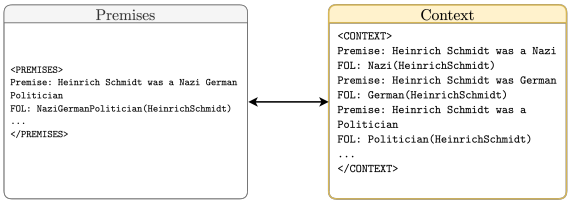
1. Categorizing Proper Names

   The fact that "Harry" is a person in the example below is common sense. We prompt the LLM to provide this information.



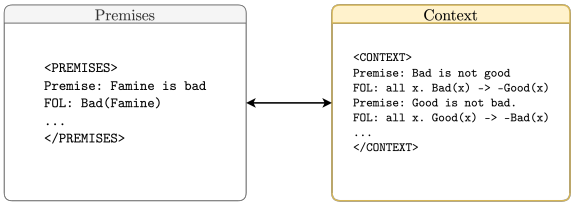
2. Separation of Properties

   In the example below "Nazi German politician" is compressed into one predicate. Thus the context generator decomposes this into predicates "Nazi", "German", and "politician"



3. Synonyms and Antonyms

   The proof checker is not aware of the semantics of the words, specifically synonym and antonym relations, thus we need to provide this information. In the example below we provide the information that "Good" and "Bad" are antonyms.



2.2 Tree of Thoughts

Inspired by Tree of Thoughts (Yao et al., 2023) we designed our pipeline based on a tree data structure as opposed to the default implementation of chain of thought prompting which is a sequential path (Figure 3).

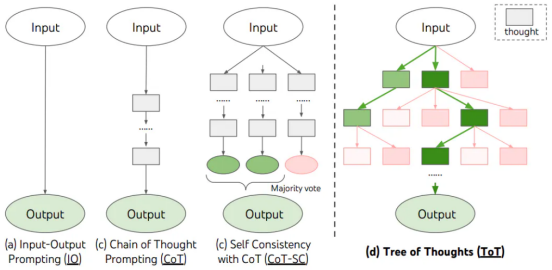


Figure 3: Tree of Thoughts (Yao et al., 2023)

There are three different types of nodes in our structure: Context-Pass-Node, Translation-Pass-Node, and Result-Pass-Node. This idea can be generalized to controlling the behavior of an LLM using a Finite State Machine (FSM). Each node in the FSM has its logic on how to interact with the LLM. The transition function between nodes is based on the result of interaction with the LLM. Additionally, each path (thought) in the tree of thought corresponds the path taken in the FSM.

This architecture allows for more efficiency and flexibility due to the following reasons:

- The modularity of logic implies flexibility in implementing different ideas in the pipeline.
- One can parallelize the task across various machines and/or API calls,
- The design implies that we can selectively avoid unnecessary branch explorations. For instance, we avoid the second pass to the LLM if based on the result of the current node the answer can be deduced with high certainty.
- This allows us to implement ideas that require a variable number of passes to the LLM. For instance, some "thoughts" might require more passes compared to others to achieve the result.

In our experiments, we defined the branching factor of the Context-Pass-Node as C and the branching factor of the Translation-Pass-Node as T.

2.3 Justification Prompting

Following (Wei et al., 2023), we ask the LLM to provide a justification for each additional sentence of context it provides. This is done in order to increase the accuracy of the LLM's translation. Without this, the LLM feels "obliged" to provide additional context. Also inspired by the compartmentalization of (Wei et al., 2023), the LLM was asked to provide context only based on a single premise, and to associate the context with that premise4. Indeed, this technique resulted in significant improvements by providing additional relevant context.

Prompting Example (Excerpt)

```
<INPUT>
    <SENTENCE>
        PREMISE:  Harry read the book "Walden" by Henry.
        FOL: Reads(Harry, Walden)
    </SENTENCE>
</INPUT>

<OUTPUT>
    <SENTENCE>
        PREMISE: Harry read the book "Walden" by Henry.
        FOL: Reads(Harry, Walden)
        CONTEXT: Walden is a book.
        FOL: Book(Walden)
        JUSTIFICATION:The text says 'the book "Walden"', so Walden is a
        book's name - this categorizes proper nouns.
    </SENTENCE>
</OUTPUT>
```

Figure 4: Example on how the context is generated for a single premise.

2.4 Overriding Errors

By observation, it became clear to the authors that the vast majority of errors arose due to either syntactic reasons (such as the arity error) or context-based translations, and so we decided to ignore any errors raised in the K-majority voting process unless all of the K votes were errors.

3 Related Works

As described earlier, LINC (Olausson and et al., 2023) is a novel approach to theorem proving. The method takes a neurosymbolic approach by combining the strengths of LLMs and FOL solvers: The LLM acts as a "semantic parser", translating the natural language premises and conclusion into FOL expressions. These FOL expressions are then fed to an automated FOL theorem prover, which solves for the truth value of the conclusion based on the premises. Our work aims to expand on the capabilities of the method by addressing the failure modes presented. The particular failure mode we focus on is missing implicit context, which we address by modifying the pipeline.

LLEMA (Wei et al., 2023) is another work on improving the reasoning capabilities of LLMs. LLEMA is an end-to-end transformer-based model that generates proofs given the description of a math problem in natural language. Though the end objective is similar in nature to ours, its methodology is largely tangential since it relies on the language model to generate a proof, whereas in LINC (Olausson and et al., 2023) the language model only serves as a parser from natural language to FOL and the SMT-Solver has the responsibility of reasoning.

Chain of Thought Prompting (Wei et al., 2023) laid the groundwork to enable LLMs to reason about the correctness of their answers. We incorporate some of the work's key ideas within our pipeline to generate high-quality context premises.

Other works which aim to address reasoning in LLMs include (Gao et al., 2023; Lyu et al., 2023; Shao et al., 2023).

4 Evaluation Criteria

We use standard-practice metrics to evaluate the performance of our method. The overall model can be interpreted as a classifier with labels which classifies whether a conclusion is "True", "False", or "Uncertain" given some premises. Our basis for evaluating our results are the same metrics in

accuracy performance that LINC is measured up against. More precisely, we evaluate the ammended pipeline over the FOLIO (Han and et al., 2022) and ProofWriter (Tafjord et al., 2021) datasets using a *confusion matrix*. These provide labelled premises and conclusions in natural language, which we can then pass through the framework and obtain the overall classifier accuracy. In particular, our contribution to improve upon the case when the framework was predicting “Uncertain" when the ground truth was either “True" or “False".

5 Results

The results of the experiment can be summarized in the following confusion matrices, see Figures 5 and 6. In particular, we see that we can produce strong results with this proposed methodology of Chain-of-Context. Throughout the experiments, we run $C = 3, T = 1$ due to limitations on computational power. Note that the K from (Olausson and et al., 2023) satisfies $K = CT$.

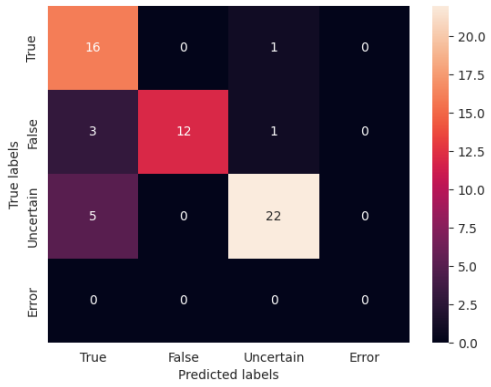


Figure 5: Confusion Matrix of results obtained majority voting, matching the true labels with an 84% accuracy on GPT-3.5-TURBO, with $C = 3, T = 1$.

6 Analysis

In this section, we analyze the results obtained in this experiment. While we cannot exactly compare with the 96.4% and 98.3 % (Olausson and et al., 2023), forGPT-3.5-TURBO and GPT-4 resp., due to differences in computational power (where they used $K = 10$), the results produced are nevertheless strong.

6.1 Successful Contexts

The suggested methodology successfully addresses a few important instances of implicit context that

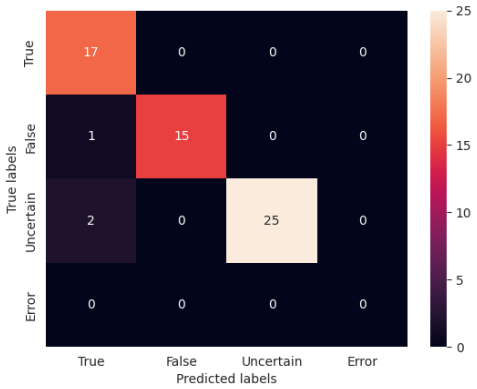


Figure 6: Confusion Matrix of results obtained majority voting, matching the true labels with an 95% accuracy on GPT-4, with $C = 3, T = 1$.

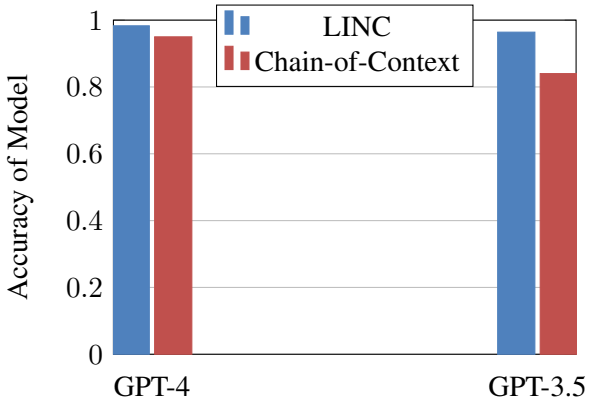


Figure 7: Comparison of LINC and Chain-of-Context. We note that this comparison is not a fair one because LINC was run on $K = 10$ while this experiment was run on $K = 3$.

arose as failure modes of (Olausson and et al., 2023). In particular, the few-shot training of language model successfully led to it adding context that clarifies the nature of antonyms, and more generally, distinct categories. For example, the LLM would successfully add premises along the lines of “If something is warm, then it is not cold” and “If something is blue, then it is not red.”

The methodology also resulted in additional contexts such.

6.2 Errors and Failure Modes

We label each data as one of the four categories of "True", "False", "Uncertain", or "Error". Below are some main reasons why we get the "Error" label.

We initially tried to generate the output in the form of JSON or XML. However, we realized that in many cases LLM would fail to generate a syn-

tactically correct JSON or XML especially when it is too long (Figure 8). It matches the intuition as the datasets that these models were trained on are mostly human-readable. As a result we changed our prompt to make our interactions with LLM human-readable.

```
["{'premises': [{'premise': 'Charlie is cold.', 'fol': '
    Cold(Charlie)'}, {'premise': 'Charlie is quiet.', '
    fol': 'Quiet(Charlie)'}, {'premise': 'Dave is blue
    .', 'fol': 'Blue(Dave)'}, {'premise': 'Dave is
    furry.', 'fol': 'Furry(Dave)'}, {'premise': 'Dave i
    ...ne: 'Quiet(x) -> Furry(x))'}, {'premise': 'If
    Charlie is smart and Charlie is cold then Charlie
    is furry.', 'fol': 'all x. (Smart(x) \& Cold(x) ->
    Furry(x))'}, {'premise': 'If something is furry
    then it is nice.', 'fol': 'all x. (Furry(x) -> Nice
    (x))'}, ...], 'conclusion': {'premise': 'Charlie is
     smart.', 'fol': 'Smart(Charlie)'}}"]
```

Figure 8: a syntactically wrong generated JSON output by LLM. It contains ... in the middle

Another major failure mode we identified is that the LLM, when asked to provide additional context, would begin *attempting* to make logical deductions of inferences. This occurs despite prompting instructions that clearly ask it not to do so, and hence suggests a need to change the prompting strategy.

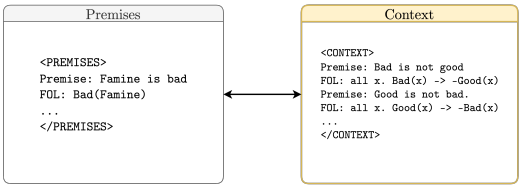


Figure 9: This is an example of context generated by the LLM. Not only does it attempt to make logical deductions, but it does so incorrectly.

This, among other things, led to a more general failure mode, which is the tendency of the method to conclude "True" when the correct answer is "False". In an attempt to add context, the LLM makes incorrect inferences (Figure 4), leading to a surprisingly high rate of occurrences where there are logically conflicting premises

7 Conclusion and Future Work

Firstly, the results we obtain fit into the larger theme of augmenting language models' reasoning abilities with symbolic logic, one which has shown promising results in the past few years. In particular, this paves the path for higher accuracy reasoning in language models. Due to the low computational complexity of our results, this also suggests the possibility of directly incorporating this into particular products of downstream applications, as we discuss in our impact statement.

Second, we acknowledge the short timeframe allocated to our project. We planned to have many intermediate deliverables in our work, with the aim of modularizing the work needed to further the research direction we are exploring.

As such, we leave to the interested reader some ideas to explore:

- The quality of the generated context relies very heavily on the prompting strategies used. We leveraged the now standard techniques described in Chain-of-Thought(Wei et al., 2023), but it would be interesting to integrate other methods such as Prompt-Breeder() to further refine the LLM querying.

- One of the major errors that our results, as well as LINC, faced, was the arity error. While this is technically beyond the scope of our context [1], it is crucial to address in the grand scheme of augmenting LLMs with reasoning abilities.

  For instance, a common problem is that one function might be used with different arities. Consider the example below:

  ```
  TEXT: The monkey eats the banana.
  FOL: eats(monkey, banana)
  TEXT: All monkeys are smart.
  FOL: all x. monkey(x) -> smart(x)
  ```

  Since the translated text has the function `monkey` with different arities, the translated FOL is invalid. In order to address this issue, one attempt was to to first prompt the large language model to list all the functions that it is going to use along with their arities, a definition in some sense. Then in order to provide the FOL translation of the text and commonsense premises, it should only use those functions with correct arity.

  Additionally, one could pass the error message generated by the proof checker as feedback to the LLM in order to refine the FOL. This can take multiple iterations.

  This was attempted at the beginning of the experiment, but led to a s urprising loss of per-

[1] Pun intended

formance and accuracy. As such, this methodology was elliminated from the experiment. While we defer the exact justifications to future work, we speculate that this led to the LLM 'committing" to a fixed set of predicates, which prevented it from making generalizable premises. We believe that it is possible to address this, for the most part, by suffixing predicates the arity, e.g. `monkey1`.

- We observed that the output of the LLM can significantly vary based on the initial prompt. As a result, as a stretch goal, we are interested in improving the prompt using newly developed techniques such as PromptBreeder (Fernando et al., 2023). PromptBreeder uses genetic algorithm to evolve the initial prompt in a way that maximizes the fitness of the prompt for the given task, and so this may help overcome the failure modes common in this field of research.

Limitations

One of the proposed methods was to include examples of failure modes of LINC in the prompt and ask the LLM to generate context based on those. A limitation of this approach is the fact that this might not cover all failure modes and additional work might be needed to identify other failure modes and include them in the prompt.

Unfortunately due to time constraints and API-key rate limiting, we were not able to run the ammended LINC pipeline on all of the tasks described in the original paper. We have limited ourselves to compare the tasks where LINC had achieved State-of-the-art results.

Impact Statement

Large Language Models (LLMs) have become crucial in advancing natural language processing and its applications, with widespread usage impacting various fields beyond computer science. However, these models have a significant drawback: the risk of generating incorrect, harmful, or offensive content. This issue limits their use in critical areas requiring reliable reasoning, such as law, medicine, and robotics. The potential risks of incorrect outputs, including severe consequences like wrongful convictions or medical errors, emphasize the need for dependable reasoning capabilities in these models(Bender et al., 2021).

This project addresses this challenge by integrating LLMs with automated theorem provers (ATPs), enhancing their ability to formalize and structure logical reasoning. Our approach leverages the complementary strengths of LLMs and classical methods. We use LLMs to assist ATPs in parsing and generating natural language, while ATPs help LLMs construct formal first-order logic (FOL) proofs. This collaboration aims to create more reliable and sound reasoning processes in LLMs, paving the way for their safer and more effective application in critical decision-making domains.

Acknowledgements

We would like to thank the instructors and teaching staff of 6.8611 for their guidance and support. We acknowledge the use of a template provided by Jordan Boyd-Graber, Naoaki Okazaki, Anna Rogers. We also acknowledge Abdurahman Sherif, Tarik Hasic, Layal Barakat, and Nurullah Giray Kuru for computational resources.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

S. Han and et al. 2022. Folio: Natural language reasoning with first-order logic. *arXiv*. Accessed: Nov. 03, 2023.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning.

T. X. Olausson and et al. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv*. Accessed: Nov. 02, 2023.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models.

Robert Stalnaker. 2002. Common ground. *Linguistics and Philosophy*, 25(5/6):701–721.

O. Tafjord, B. Dalvi, and P. Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

A Dataset Preprocessing

As in (Olausson and et al., 2023), we use the publicly available FOLIO dataset, and build on their codebase, which already pre-processed the data. We build our own tree-based methods to parallelize the search, given the limited computational resources we have access to. Our lightly-documented codebase can be found at `https://github.com/Shayan-P/Chain-of-Context`.

B Prompts

We use two passes to the LLM with different prompts. The first pass if for translating natural language to formal language and the second pass is to add context to formal statements. The prompts that we used in each of those passes are shown below.

B.0.1 Prompt: Translation to FOL

```
The following is a first-order logic (FOL) problem.
The problem is to determine whether the conclusion follows
      from the premises.
The premises are given in the form of a set of first-order
      logic sentences.
The conclusion is given in the form of a single first-order
      logic sentence.
The task is to translate each of the premises and
      conclusions into FOL expressions, so that the
      expressions can be evaluated by a theorem solver to
      determine whether the conclusion follows from the
      premises.
Expressions should be adhere to the format of the Python
      NLTK package logic module.

<INPUT>

<PREMISES>
All dispensable things are environment-friendly.
All woodware is dispensable.
All paper is woodware.
No good things are bad.
All environment-friendly things are good.
A worksheet is either paper or is environment-friendly.
</PREMISES>
<CONCLUSION>
A worksheet is not dispensable.
</CONCLUSION>

</INPUT>
<OUTPUT>

<PREMISES>

PREMISE: All dispensable things are environment-friendly.
FOL: all x. (Dispensable(x) -> EnvironmentFriendly(x))


PREMISE: All woodware is dispensable.
FOL: all x. (Woodware(x) -> Dispensable(x))


PREMISE: All paper is woodware.
FOL: all x. (Paper(x) -> Woodware(x))


PREMISE: No good things are bad.
FOL: all x. (Good(x) -> -Bad(x))


PREMISE: All environment-friendly things are good.
FOL: all x. (EnvironmentFriendly(x) -> Good(x))


PREMISE: A worksheet is either paper or is environment-
      friendly.
FOL: ((Paper(Worksheet) & -EnvironmentFriendly(Worksheet)) |
       (-Paper(Worksheet) & EnvironmentFriendly(Worksheet)))

</PREMISES>
<CONCLUSION>
```

```
PREMISE: A worksheet is not dispensable.
FOL: -Dispensable(Worksheet)

<CONCLUSION>

</OUTPUT>


<INPUT>

<PREMISES>
A La Liga soccer team ranks higher than another if it
      receives more points.
If two La Liga soccer teams recieve the same points, the
      team which recieves more points from the games between
      the two teams ranks higher.
Real Madrid and Barcelona are both La Liga soccer teams.
In La Liga 2021-2022, Real Madrid recieves 86 points and
      Barcelon recieves 73 points.
In La Liga 2021-2022, Real Madrid and Barcelona both recieve
       3 points from the games between them.
</PREMISES>
<CONCLUSION>
In La Liga 2021-2022, Real Madrid ranks higher than
      Barcelona.
</CONCLUSION>

</INPUT>
<OUTPUT>

<PREMISES>

PREMISE: A La Liga soccer team ranks higher than another if
      it receives more points.
FOL: all x. all y. (LaLiga(x) & LaLiga(y) & MorePoints(x, y)
      -> HigherRank(x, y))


PREMISE: If two La Liga soccer teams recieve the same points
      , the team which recieves more points from the games
      between the two teams ranks higher.
FOL: all x. all y. (LaLiga(x) & LaLiga(y) & -MorePoints(x, y
      ) & -MorePoints(y, x) & MorePointsInGameBetween(x, y)
      -> HigherRank(x, y))


PREMISE: Real Madrid and Barcelona are both La Liga soccer
      teams.
FOL: LaLiga(RealMadrid) & LaLiga(Barcelona)


PREMISE: In La Liga 2021-2022, Real Madrid recieves 86
      points and Barcelon recieves 73 points.
FOL: MorePoints(RealMadrid, Barcelona)


PREMISE: In La Liga 2021-2022, Real Madrid and Barcelona
      both recieve 3 points from the games between them.
FOL: -MorePointsInGameBetween(RealMadrid, Barcelona) & -
      MorePointsInGameBetween(Barcelona, RealMadrid)

</PREMISES>
<CONCLUSION>

PREMISE: In La Liga 2021-2022, Real Madrid ranks higher than
       Barcelona.
FOL: HigherRank(RealMadrid, Barcelona)

<CONCLUSION>

</OUTPUT>


<INPUT>

<PREMISES>
Charlie is cold.
Charlie is furry.
Charlie is kind.
Charlie is nice.
Charlie is red.
Charlie is rough.
Dave is red.
Dave is rough.
Fiona is rough.
Harry is kind.
Harry is rough.
Red things are nice.
All nice things are cold.
Furry things are kind.
If something is cold and rough then it is white.
If Fiona is furry then Fiona is kind.
Rough, kind things are furry.
White things are kind.
</PREMISES>
<CONCLUSION>
Fiona is white.
</CONCLUSION>
</INPUT>
<OUTPUT>
```

B.0.2 Prompt: Adding Context

```
The following is a first-order logic (FOL) problem.
The problem is to determine whether the conclusion follows
      from the premises.
The premises are given in the form of a set of first-order
      logic sentences.
The conclusion is given in the form of a single first-order
      logic sentence.
The task is to translate each of the premises and
      conclusions into FOL expressions, so that the
      expressions can be evaluated by a theorem solver to
      determine whether the conclusion follows from the
      premises.
Expressions should be adhere to the format of the Python
      NLTK package logic module.



<INPUT>

<PREMISES>
All dispensable things are environment-friendly.
All woodware is dispensable.
All paper is woodware.
No good things are bad.
All environment-friendly things are good.
A worksheet is either paper or is environment-friendly.
</PREMISES>
<CONCLUSION>
A worksheet is not dispensable.
</CONCLUSION>

</INPUT>
<OUTPUT>

<PREMISES>

PREMISE: All dispensable things are environment-friendly.
FOL: all x. (Dispensable(x) -> EnvironmentFriendly(x))


PREMISE: All woodware is dispensable.
FOL: all x. (Woodware(x) -> Dispensable(x))


PREMISE: All paper is woodware.
FOL: all x. (Paper(x) -> Woodware(x))


PREMISE: No good things are bad.
FOL: all x. (Good(x) -> -Bad(x))


PREMISE: All environment-friendly things are good.
FOL: all x. (EnvironmentFriendly(x) -> Good(x))


PREMISE: A worksheet is either paper or is environment-
      friendly.
FOL: ((Paper(Worksheet) & -EnvironmentFriendly(Worksheet)) |
       (-Paper(Worksheet) & EnvironmentFriendly(Worksheet)))

</PREMISES>
<CONCLUSION>

PREMISE: A worksheet is not dispensable.
FOL: -Dispensable(Worksheet)

<CONCLUSION>

</OUTPUT>


<INPUT>

<PREMISES>
A La Liga soccer team ranks higher than another if it
      receives more points.
If two La Liga soccer teams recieve the same points, the
      team which recieves more points from the games between
      the two teams ranks higher.
Real Madrid and Barcelona are both La Liga soccer teams.
In La Liga 2021-2022, Real Madrid recieves 86 points and
      Barcelon recieves 73 points.
In La Liga 2021-2022, Real Madrid and Barcelona both recieve
       3 points from the games between them.
</PREMISES>
<CONCLUSION>
In La Liga 2021-2022, Real Madrid ranks higher than
      Barcelona.
</CONCLUSION>
```

</INPUT>
<OUTPUT>

<PREMISES>

PREMISE: A La Liga soccer team ranks higher than another if it receives more points.
FOL: all x. all y. (LaLiga(x) & LaLiga(y) & MorePoints(x, y) -> HigherRank(x, y))

PREMISE: If two La Liga soccer teams recieve the same points, the team which recieves more points from the games between the two teams ranks higher.
FOL: all x. all y. (LaLiga(x) & LaLiga(y) & -MorePoints(x, y) & -MorePoints(y, x) & MorePointsInGameBetween(x, y) -> HigherRank(x, y))

PREMISE: Real Madrid and Barcelona are both La Liga soccer teams.
FOL: LaLiga(RealMadrid) & LaLiga(Barcelona)

PREMISE: In La Liga 2021-2022, Real Madrid recieves 86 points and Barcelon recieves 73 points.
FOL: MorePoints(RealMadrid, Barcelona)

PREMISE: In La Liga 2021-2022, Real Madrid and Barcelona both recieve 3 points from the games between them.
FOL: -MorePointsInGameBetween(RealMadrid, Barcelona) & -MorePointsInGameBetween(Barcelona, RealMadrid)

</PREMISES>
<CONCLUSION>

PREMISE: In La Liga 2021-2022, Real Madrid ranks higher than Barcelona.
FOL: HigherRank(RealMadrid, Barcelona)

<CONCLUSION>

</OUTPUT>

<INPUT>

<PREMISES>
Anne is rough.
Charlie is red.
Charlie is rough.
Erin is blue.
Erin is white.
Harry is blue.
Harry is white.
Blue, nice things are white.
If Charlie is round and Charlie is rough then Charlie is young.
All nice, white things are rough.
Round things are rough.
If something is young then it is red.
If something is nice then it is round.
All rough, blue things are round.
Young things are nice.
If Charlie is red then Charlie is blue.
</PREMISES>
<CONCLUSION>
Harry is rough.
</CONCLUSION>

</INPUT>
<OUTPUT>