ENHANCING AND EVALUATING LOGICAL REASONING ABILITIES OF LARGE LANGUAGE MODELS

Shujie Deng
University of Toronto & Vector Institute
shujie.deng@mail.utoronto.ca

Honghua Dong
University of Toronto & Vector Institute
honghuad@cs.toronto.edu

Xujie Si
University of Toronto & Vector Institute
six@cs.toronto.edu

ABSTRACT

Despite recent advancements in Large Language Models (LLMs), challenges persist in their ability to process complex logical rules. Previous work of Logic-LM (Pan et al., 2023) integrates LLMs with separate symbolic solvers for various reasoning tasks. While it is effective, it is hard to scale across different tasks. This paper introduces an innovative framework that unifies the integration of LLMs with a Z3 symbolic solver to solve various reasoning tasks. The integration is complemented by an additional *Self-Refinement Module* to enhance the reliability of code generation of LLM. We evaluated the LLM's performance on four diverse datasets - ProntoQA, ProofWriter, FOLIO, and Logical Deduction - covering a range of deductive, analytical and first-order-logic(FOL) reasoning tasks. Our framework demonstrates significant improvements, outperforming Logic-LM by 4.86% and 7.82% on GPT-3.5-Turbo and GPT-4 models, respectively. Through an analysis of failure cases, we identify several limitations in LLM translation, such as misinterpretation of relationships, literal translation lacking contextual understanding, and misapplication of logical structures.

1 INTRODUCTION

Large Language Models(LLMs) have achieved significant breakthroughs in the field of machine learning, demonstrating remarkable emergent abilities (Wei et al., 2022a) across various tasks. Central to these abilities is the logical reasoning ability, which is pivotal in defining the overall quality of LLM-generated content. Logical reasoning is instrumental in enhancing complex problem-solving capabilities, ensuring reliability and accuracy. Furthermore, it plays a pivotal role in the model's capacity for learning and adaptation, which are essential for tackling a range of tasks effectively.

Despite recent remarkable advancements, LLMs are not without their limitations. Some notable challenges are that LLMs cannot plan ahead, and they fall short in effectively interpreting longer texts and complex logical rules, leading to hallucination in their outputs (Huang et al., 2023; Valmeekam et al., 2024). Therefore, enhancing logical reasoning abilities in LLMs becomes a meaningful and challenging question.

To address this challenge, we take inspiration from the dual process theory (Daniel, 2017). The theory illustrates two distinct systems of human cognition: *System 1* thinking, which is rapid and instinctive, and *System 2* thinking, which is slower but more deliberate and logical. We observe that LLMs are good at making fast and intuitive inferences, and symbolic solvers, particularly the Z3 symbolic solver (De Moura and Bjørner, 2008), are good at making logical and accurate predictions. The Z3 symbolic solver is especially good at tackling various logical reasoning challenges, including deductive reasoning, first-order logic reasoning and constraint satisfaction problems. Leveraging these insights, we propose a novel framework that integrates LLMs with a unified Z3 solver across diverse tasks. In this framework, the LLM translates natural language problems into executable neural-symbolic codes, which are then processed by the Z3 solver to yield outputs. Additionally, we incorporate *Self-Refinement Module*, an iterative feedback module, which passes the unsuccessful

execution results from the Z3 solver back to LLM, thereby ensuring the LLM's translation reliability and correctness.

Finally, we conducted a comparative analysis of our framework's performance with that of Logic-LM (Pan et al., 2023) using GPT-3.5 and GPT-4 (Brown et al., 2020; OpenAI, 2023) across four logical reasoning datasets, using few-shot prompting. The results show a notable average improvement of 4.86% and 7.82% on GPT-3.5-Turbo and GPT-4 respectively. Additionally, we conducted an in-depth study of the failure cases when LLM fails to generate correct output and analyze their causes. This investigation revealed several limitations in LLM translation capabilities, highlighting areas that require further enhancement.

In summary, our main contributions are:

- We propose a novel framework that integrates LLM with a unified Z3 symbolic solver with a *Self-Refinement Module* to improve the logical reasoning ability of LLM.
- We conduct a comprehensive analysis of failure cases in LLM translation and provide valuable insights into the limitations and areas for improvement of reasoning of large language models.

2 METHODOLOGY

Our framework consists of three modules, an LLM translation module, a z3 solver module, and an interactive feedback module, as is shown in *Figure 1*.

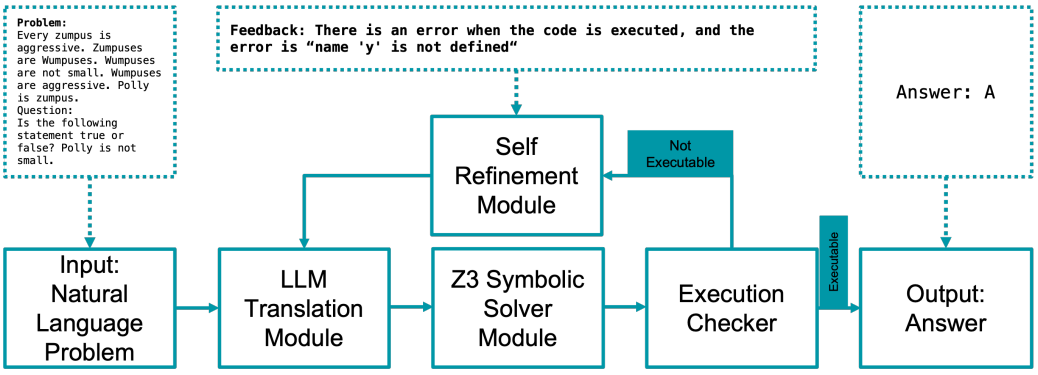

Figure 1: **General architecture of the framework.** Each solid box represents a module in the framework, and each dashed line box represents a ProntoQA example of that module for clarification.

LLM Translation Module In addressing natural language problems (see Section 3.1 for more details about datasets), we first design an *LLM Translation Module* to translate such problems into executable Z3 code compatible with Python. It employs a combination of instruction prompts and user prompts, distinct in their functions, to guide the language model in generating Z3 code that is both syntactically and semantically correct. The design of these prompts is meticulously oriented to ensure three key outcomes: 1) The translated output code is executable. 2) The code execution result is one of the provided options in the input problem. 3) The semantic meaning of code matches the semantic meaning of the natural language problem. The module utilizes two-shot prompting approach, which involves guided step-by-step instructions followed by Chain-of-Thought (Wei et al., 2022b) prompting strategy. Examples of the prompts can be seen in Appendix A.

Z3 Symbolic Solver Module Z3 solver (De Moura and Bjørner, 2008) is a high-performance automated solver, belonging to the family of tools known as Satisfiability Modulo Theories(SMT) solvers. The Z3 solver is designed to check the satisfiability of logical formulas across a range of theories, including arithmetic, uninterpreted functions, and more. It is capable of determining whether a given set of constraints can be satisfied and, if so, can provide a model that satisfies those

constraints. In our framework, The *Z3 Symbolic Solver Module* executes the output code generated by *LLM Translation Module* to obtain the final answer, which are options in multiple-choice questions.

Self-Refinement Module The *Self-refinement Module* is specifically designed to address cases where the Z3 code generated by *LLM Translation Module* is not executable. In such cases, the module saves the error message and feeds it back to the *LLM Translation Module*. The error message serves as an additional prompt, guiding the language model to regenerate the Z3 code. Subsequently, the revised code is passed again to the *Z3 Symbolic Solver Module* for execution. This process is iterative, continuing until the code becomes executable or until a predefined maximum number of iterations is reached. This module plays a crucial role in enhancing the reliability of our framework by ensuring the successful execution of generated code.

3 Experiments and Evaluations

3.1 Datasets

We test the performance of our framework on four tasks, ProntoQA, ProofWriter, FOLIO and LogicalDeduction (Saparov and He, 2022; Tafjord et al., 2020; Han et al., 2022; Srivastava et al., 2022) that span deductive reasoning, first-order logic(FOL) reasoning and constraint satisfaction reasoning(CSP) tasks, as is shown in Table 1.

Dataset	Reasoning Tasks	Size	Output Options
ProntoQA	Deductive Reasoning	500	A, B
ProofWriter	Deductive Reasoning	4200	A, B, C
FOLIO	First Order Logic Reasoning	1208	A, B, C
LogicalDeduction	Constraint Satisfaction Reasoning	4500	A, B, C, D, E, F, G, H, I

Table 1: Datasets overview

3.2 Results

From results shown in Table 2, we have the following observations.
First, we can see that the accuracy of our framework substantially outperforms Logic-LM in all four datasets in most cases except the accuracy on GPT-3.5-Turbo on FOLIO. With GPT-3.5-Turbo, we improved the average accuracy by 4.86%; With GPT-4, we improved the average accuracy by 7.82%. The result is due to the addition of the self-refinement module and the increasing power of Z3 solver compared to other symbolic solvers.
Next, on GPT-4, the performance of our framework on dataset ProntoQA, ProofWriter and Logical Deduction is almost saturated, with accuracy of 94%, 96% and 98% respectively.
Finally, the accuracy of FOLIO, a first-order logic task is the lowest among all logical reasoning tasks. This is mostly because the tasks in FOLIO are more diverse in terms of context, and they are more complex in terms of logic operations.

Dataset	Model Name			
	GPT-3.5-Turbo		GPT-4	
	Logic-LM	Our Framework	Logic-LM	Our Framework
ProntoQA	93.20%	**96.00%**	93.60%	**94.00%**
ProofWriter	70.11%	**86.00%**	79.33%	**96.00%**
FOLIO	**54.60%**	45.00%	74.50%	**80.00%**
Logical Deduction	67.66%	**78.00%**	89.29%	**98.00**%

Table 2: **Accuracy for Logic-LM and our framework of GPT-3.5-Turbo and GPT-4 on 4 datasets** For our framework, we use a subset of 50 randomly selected samples for each dataset and each language model.

3.3 FAILURE CASES OF LLM TRANSLATION MODULE

Appendix B) shows a detailed analysis of typical failure cases of *LLM Transaltion Module*.By studying these typical failure cases, we find that the LLM translation suffers from the challenges of misinterpretation of relationships, literal translation without contextual understanding, and logical structure misapplication.

4 RELATED WORK

Evaluation of LLM's Logical Reasoning Ability With the invention of LLMs like ChatGPT, various logical evaluations have been conducted on these LLMs((Huang and Chang, 2022; Qiao et al., 2022). Notably, Liu et al. (Liu et al., 2023) found that the performance of GPT-4 and ChatGPT drops significantly when handling newly released and out-of-distribution datasets; Huang et al. (Huang and Chang, 2022) concluded that LLMs are still unskilled at complex reasoning tasks involving implicature. In our framework, we dealt with the limitations of LLM's logical reasoning by incorporating it with a symbolic solver.

Synthetic reasoning datasets Synthetic datasets on logical reasoning have been created to fully benchmark the logical reasoning ability of LLMs. *LogicBench* (Anonymous, 2023) introduced a natural language question-answering dataset encompassing 25 different reasoning patterns spanning over propositional, first-order and non-monotonic logics and benchmarked 5 Language Models(LMs) on their datasets. While their benchmarking focuses only on their synthetic generated datasets, we benchmarked on 4 datasets. When designing prompts for LLM Translation Module, we took the insights from their discovery that LLM experienced challenges in understanding negations and longer inference rules.

In-Context learning Our work on designing LLM translation module is related to in-context learning so that prompts are designed to fully achieve LLM's capacity. Various methodologies are created to improve in-context learning. Chain-of-thought(CoT) prompting (Wei et al., 2022b) decomposes complex problems into multiple steps, which significantly improves the ability of large language models to perform complex reasoning. Least-to-most prompting (Zhou et al., 2022) followed up the work of CoT by breaking down a complex problem into a series of simpler subproblems and then solving them in order, with each subproblem being facilitated by the answers obtained from previously solved problems. Tree-of-thought(ToT) prompting (Yao et al., 2023) further generalized these two prompting techniques by considering multiple different reasoning paths and self-evaluating choices to decide the next course of action and make global decisions when necessary. Adopting these in-context learning techniques, we added step-by-step instructions for translation and enforced several requirements of the output of translation to be met in instruction prompts.

Incoprating LLM with tools To augment complex reasoning capabilities, integrating external reasoning tools with LLMs has been a recent approach, such as Toolformer (Schick et al., 2023), Chameleon (Chartrand and Bargh, 1999) and SatLM (Ye et al., 2023). Notably, SatLM generated a declarative task specification and leveraged off-the-shelf automated theorem prover to derive the final answer. Our framework presents several key differences from SatLM: 1) Our framework consists of a self-refinement module designed to iteratively improve the LLM's translation performance. 2) The evaluation benchmarks we employ are different from those used in the assessment of SatLM, offering unique insights and results.

5 CONCLUSIONS AND FUTURE WORK

In conclusion, our proposed framework integrates Large Language Models(LLMs) with a Z3 symbolic solver and a self-refinement module, enhancing logical reasoning capabilities and achieving accuracy improvements of 4.86% and 7.82% on GPT-3.5-Turbo and GPT-4, respectively, across four logical reasoning datasets. Our analysis of failure cases revealed limitations in LLM translation, including misinterpretation, lack of contextual understanding, and misapplication of logical structures. We also observed near-peak performance of GPT-4 on several datasets, indicating a need for more complex

and diverse challenges, potentially through synthetic datasets similar to LogicBench (Anonymous, 2023).

Additionally, we recognize the necessity of further verifying the validity of correct predictions, as current multiple-choice formats may mask translation inaccuracies. Future enhancements could include an LLM self-checker module for pre-validation of translations and the use of natural language responses to reduce the likelihood of false positives.

6 ACKNOWLEDGEMENTS

We thank Zhaoyu Li, Saifei Liao, Haokun Liu, Adrian Zhao and Ruiyu Wang for helpful insights and discussions on the paper draft. We thank Allen Geng, Zenan Li and Yu Yang for their assistance at the initial stage of the project.

REFERENCES

Anonymous (2023). Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Submitted to The Twelfth International Conference on Learning Representations*. under review.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chartrand, T. L. and Bargh, J. A. (1999). The chameleon effect: The perception–behavior link and social interaction. *Journal of personality and social psychology*, 76(6):893.

Daniel, K. (2017). *Thinking, fast and slow*.

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

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Benson, L., Sun, L., Zubova, E., Qiao, Y., Burtell, M., et al. (2022). Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Huang, J. and Chang, K. C.-C. (2022). Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. (2023). Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Liu, H., Ning, R., Teng, Z., Liu, J., Zhou, Q., and Zhang, Y. (2023). Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

OpenAI, R. (2023). Gpt-4 technical report. *arXiv*, pages 2303–08774.

Pan, L., Albalak, A., Wang, X., and Wang, W. Y. (2023). Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*.

Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., and Chen, H. (2022). Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Saparov, A. and He, H. (2022). Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Tafjord, O., Mishra, B. D., and Clark, P. (2020). Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. (2024). Planbench: an extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022a). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022b). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Ye, X., Chen, Q., Dillig, I., and Durrett, G. (2023). Satlm: Satisfiability-aided language models using declarative prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. (2022). Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A PROMPTS FOR LLM TRANSLATION MODULE

User Prompt for ProntoQA:

````
Task Description: You are given a problem description and a question. The task is to write a python script which includes:
1) Define all variables for all entities in the problem. You should write a comment indicating which sentence the entity is in,and if the entity has been defined before, also write a comment indicating it has been defined and not define the entity again.
2) Create a solver instance
3) Parse the problem into relationships based on the defined entities, you should only use `Implies` and `Not` in this part. You should write a comment indicating which sentence the relationship is in.
4) Create statements to be checked. You shoud write a comment indicating the statement.
5) Check if the solver can find a model that satisfies the conditions, if true, return A, if false, return B

Following is an example to follow:
Problem:
Every zumpus is aggressive. Zumpuses are Wumpuses. Wumpuses are not small. Wumpuses are aggressive. Polly is zumpus.
Question:
Is the following statement true or false? Polly is not small.
Options:
A, B

```python
from z3 import *

# Define boolean variables for all entities
# Every zumpus is aggressive
zumpus = Bool("zumpus")
aggressive = Bool('aggressive')

# Zumpuses are Wumpuses
# zumpus has been defined before
wumpus = Bool('wumpus')

# Wumpuses are not small.
# Wumpuses has been defined before
small = Bool("small")

# Wumpuses are aggressive.
# Wumpuses has been defined before
# Aggressive has been defined before

# Polly is zumpus.
polly = Bool("polly")
# Zumpuses has been defined before

# Create a solver instance
solver = Solver()

# Parse the problem into relationships
# Every zumpus is aggressive
solver.add(Implies(zumpus, aggressive))
# Zumpuses are Wumpuses
solver.add(Implies(zumpus, wumpus))
# Wumpuses are not small
solver.add(Implies(wumpus, Not(small)))
# Wumpuses are aggressive.
solver.add(Implies(wumpus, aggressive))
# Polly is zumpus
````
````

```
solver.add(Implies(polly, zumpus))

# Create facts in the problem
solver.add(polly)

# Create statements to be checked
# Polly is not small.
solver.add(polly, Not(small))

# Check if the solver can find a model that satisfies the conditions
if solver.check() == sat:
    print("A")  # The statement is true
else:
    print("B")  # The statement is false
```
>>
[Problem]:
%s
[Question]:
%s
[Options]:
%s
"""

Instruction Prompt for ProntoQA:

```
You should write in the format as comment in python script as follows:
Define boolean variables for all entities: <the entities you generate>
Create a solver instance: `solver = Solver()`
Parse the problem into relationships: <Relationships you generate>
Create statements to be checked: <Statements to be checked>
Check if satisfy the condition: print A if the statement is true and B if
     the statement is false

You must meet the following requirements:
1. The output should be a python script of the format ```python <code
    generate> ```
2. All boolen variables should be defined, and you should only define
    each variable once, not multiple times.
3. Number of Relationships you generate should be the same as number of
    sentences in the [Problem](A sentence is defined as ending with ".").
```

B FAIURE CASES STUDIES OF THE LLM TRANSLATION MODULE

In this section, we discussed some typical failure cases of the LLM translation module and their causes.

FAILURE CASE 1: FOLIO_10

```
Context:
There are six types of wild turkeys: Eastern wild turkey, Osceola wild
    turkey, Gould\u2019s wild turkey, Merriam\u2019s wild turkey, Rio
    Grande wild turkey, and Ocellated wild turkey. Tom is not an Eastern
    wild turkey. Tom is not an Osceola wild turkey. Tom is also not a
    Gould's wild turkey, or a Merriam's wild turkey, or a Rio Grande wild
     turkey. Tom is a wild turkey.

Translation:
# Define all facts
facts = [
    # Tom is not an Eastern wild turkey.
    Not(EasternWildTurkey(Tom)),
    # Tom is not an Osceola wild turkey.
    Not(OsceolaWildTurkey(Tom)),
    # Tom is not a Gould's wild turkey.
    Not(GouldsWildTurkey(Tom)),
    # Tom is not a Merriam's wild turkey.
    Not(MerriamsWildTurkey(Tom)),
    # Tom is not a Rio Grande wild turkey.
    Not(RioGrandeWildTurkey(Tom)),
    # Tom is a wild turkey.
    WildTurkey(Tom)
]
```

Analysis:
Misalignment with Contextual Intent: The large language model can not remember context across sentences. In the original context, we are given a list of six types of wild turkeys and a series of statements about what Tom is not. The logical intent behind presenting this information deduces that Tom must be the one type of turkey not negated by the statement, which in this case is the Ocellated wild turkey. However, the LLM translation fails to align with this intent since it actually translation the negations(i.e. what Tom is not), but it does not translate the fact that there are indeed only 6 types of turkeys.
Cause: This issue could be due to the model's focus on the literal translation of statements rather than interpreting the broader context or narrative behind the text. LLMs sometimes miss the "bigger picture" or overarching narrative in a given text.

FAILURE CASE 2: FOLIO_5

```
Context:
Employees will either have lunch in the company or have lunch at home.


Translation:
# Employees will either have lunch in the company or have lunch at home.
ForAll([x], Xor(HaveLunchInCompany(x), Not(HaveLunchInCompany(x))))
```

Analysis:
Over-simplification: The original natural language sentence was meant to express that employees have two exclusive options for lunch: either they have lunch in the company, or they have lunch at home. However, the LLM failed to specify the alternative option(having lunch at home) explicitly. The translation oversimplified the scenario by reducing the option "having lunch at home" to the negation of "having lunch in the company".
Cause: The model did not fully grasp the context that "not having lunch in the company" does not automatically imply "having lunch at home". There might be a gap in the model's ability to formulate

logical expressions that accurately reflect the subtleties of natural language statements, especially when it involves translating informal or implied meanings into formal logic.

FAILURE CASE 3: FOLIO_7

```
Context:
Monkeypox is an infectious disease caused by the monkeypox virus.
    Monkeypox virus can occur in certain animals, including humans.
    Humans are mammals. Mammals are animals. Symptons of Monkeypox
    include fever, headache, muscle pains, feeling tired, and so on.
    People feel tired when they get a glu.

Translation:
# Monkeypox is an infectious disease caused by the monkeypox virus.
ForAll([x], Implies(Monkeypox(x), InfectiousDisease(x))),
# Monkeypox virus can occur in certain animals, including humans.
ForAll([x], Implies(MonkeypoxVirus(x), Animal(x))),
# Humans are mammals.
ForAll([x], Implies(Mammal(x), Animal(x))),
# Mammals are animals.
ForAll([x], Implies(Animal(x), Mammal(x))),
# Symptoms of Monkeypox include fever, headache, muscle pains, feeling
    tired, and so on.
ForAll([x], Implies(Symptoms(x), And(Fever(x), Headache(x), MusclePains(x
    ), FeelingTired(x)))),
# People feel tired when they get a glu.
ForAll([x], Implies(FeelingTired(x), Glu(x)))
```

Analysis:
Misinterpretation of Relationships: For the second sentence, the translation implies that anything with the monkeypox virus is an animal, it misinterpreted the relationship between the virus and animals. It should represent the potential presence of the virus in animals, not define the virus as an animal. And for the fifth sentence, the translated sentence implied that if something is a symptom of monkeypox, it must be all of these symptoms simultaneously, which is incorrect, The correct translation should individually link each symptom to monkeypox. The last sentence's translation is also incorrect. It wrongly correlates feeling tired exclusively with having "glu", which reverses the cause-and-effect relationship.
Cause: There is a misapplication of logical structures, such as implications and conjunctions, leading to incorrect or nonsensical statements. Moreover, the LLM failed to correctly interpret and translate the relationships and hierarchies(eg. humans as mammals, mammals as animals), and it seems to have translated the statement too literally without understanding the broader context or the nature of biological classification and symptoms. Summarizing the above three typical failure cases, we concluded that the LLM translation models suffer from the challenges of misinterpretation of relationships, literal translation without contextual understanding, and logical structure misapplication.

C ALGORITHM FOR THE FRAMEWORK

We provide the following pseudo-code *Algorithm 1* for the implementation of our framework.

Algorithm 1 Framework Code

```
Import Datasets D
for all data in D do
    i ← 0
    Executable ← False
    while i < num_iterations & not Executable do
        prepare prompt p
        code = translate(data.context, data.problem, prompt)
        Executable, output = code.execute()
        if not Executable then
            prompt.append(output)        ▷ add the error message to the prompt for next iteration
        end if
    end while
    Add output to final evaluation
end for
Final Evaluation
```