A Study on Translating Logical Specifications to Natural Language using Large Language Models

Antony Brijesh[1], Yunja Choi[2]
[1] School of Electronics Engineering, Kyungpook National University
[2] School of Computer Science Engineering, Kyungpook National University
brijeshantonio13@gmail.com,yuchoi76@knu.ac.kr

***Abstract*-**Requirements specifications are necessary and important in system development, but they often do not exist or are incomplete. One approach to address this is automatically generating formal logical specifications from source code. However, these generated specifications are typically difficult for non-technical stakeholders to understand, hindering verification. To bridge this gap, this study investigates translating logical specifications into natural language using Large Language Models (LLMs).

***Index Terms*-** Formal Specifications, Large Language Models (LLMs), Natural Language Translation, Program Verification, SMT Solvers

1. INTRODUCTION AND MOTIVATION

Requirement specifications are vital for reliable software, particularly in safety-critical systems, yet they are often missing or incomplete, hindering verification and creating technical debt. [1]Automatically generated logical specifications from code can help, but their formal, symbolic nature (see Figure 1) makes them opaque to non-technical stakeholders.

```
((dw.is_DiscPresent == IN_FF) && !(inp.DiscEject)
&& !((!(rtY.MechCmd + 2 == PLAY) && !(rtY.MechCmd + 2
== REW) && !inp.DiscEject && !(rtY.MechCmd + 2 == FF)
&& !(rtY.MechCmd + 2 == EMPTY) && !(rtY.MechCmd + 2 ==
DISCINSERT))) && !((rtY.MechCmd + 2 == REW))
&& !((rtY.MechCmd + 2 == PLAY))) ->
(rtDW.is_DiscPresent == IN_FF)
```

Figure 1 : logical Statement

This opacity risks misinterpretation, leading to requirement misalignments that formal verification tools alone cannot detect. Translating these specifications into Natural Language (NL) offers a path for crucial human-centric verification by domain experts. While Large Language Models (LLMs) excel at many NL tasks [5], our investigation shows they struggle to reliably translate complex formal logic (e.g., involving double negations or deep nesting), potentially introducing critical inaccuracies. Therefore, direct LLM translation is insufficient for this task due to trustworthiness concerns. This study focuses on the essential role of verified automation. We propose and evaluate an automated pipeline specifically designed to handle formal specification intricacies. Key steps include preprocessing specifications, identifying complex structures for targeted handling, and crucially, incorporating formal verification (using SMT solvers) to guarantee the logical integrity of any transformations before final NL generation by the LLM. Our work concentrates on establishing this robust, automation-centric methodology to reliably bridge the communication gap between formal specifications and human understanding.

2. LIMITATIONS OF DIRECT LLM TRANSLATION

The logical specifications for this study originate from prior work [1] and were automatically generated from the C source code of seven diverse embedded system programs (e.g., CD Player/Radio, Launch Abort System, Traffic Intersection). Our dataset comprises 74 distinct specifications from these programs, with 6-22 per program.

In our initial phase, we investigated direct LLM translation of these formal specifications into natural language, exploring various methods with different LLMs and prompting strategies (from basic to Chain-of-Thought (CoT) [2, 3]). Despite using advanced techniques, a critical limitation emerged across all approaches: LLMs consistently struggled to reliably translate a significant portion (~31 of 74) of the specifications, often yielding semantically incorrect or incomplete outputs.

Analysis showed these failures predominantly involved specifications with high logical complexity, including known challenges like double negations, excessive length, or intricate nesting and semantic relationships. The inability of even sophisticated direct translation methods to reliably handle these complex specifications underscored the risks of relying solely on LLMs for this sensitive task. This finding was the primary motivation for developing the automated, verified pipeline detailed in Section 3. This pipeline aims to systematically overcome these observed difficulties, using features like complexity analysis and formal verification to ensure logical integrity before the final translation step.

3. AUTOMATION

Motivated by the significant challenges encountered when attempting direct LLM translation of complex logical specifications (as detailed in Section 2), we designed and implemented an automated pipeline. This pipeline, developed in Python and utilizing the Gemini API (illustrated in Figure 2),

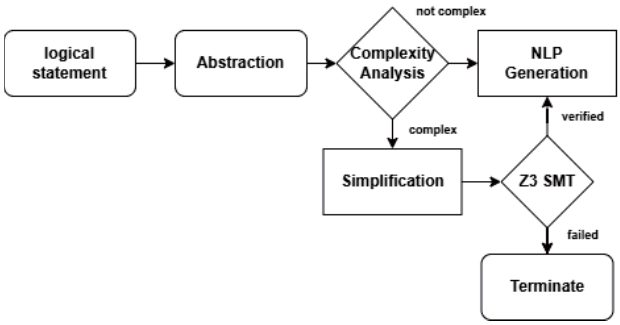

Figure 2: Automation pipeline

aims specifically to overcome the identified limitations by systematically managing specification complexity and

rigorously ensuring logical integrity before the final natural language generation step.

The pipeline workflow includes:

1. Input Abstraction: Replaces lengthy terms with placeholders to reduce tokens and standardize input structure.
2. Complexity Analysis & Routing: Uses heuristics (detecting double negations, length, etc.) to identify complex specifications. Simpler statements proceed directly to translation; complex ones enter a simplification/verification path.
3. Targeted Simplification (for Complex): Employs the Gemini API to attempt logical simplification of complex statements.
4. Mandatory Z3 Verification: Formally verifies the LLM's simplified output against the original specification for logical equivalence using the Z3 SMT solver. This is a crucial integrity check.
5. Conditional Translation: Only specifications initially deemed simple or those whose simplification is successfully Z3-verified are passed to the LLM for final natural language translation.

This verified, conditional approach systematically manages complexity and safeguards logical integrity, addressing the limitations of direct translation.
Limitation & Benefits: Integrating targeted LLM simplification with mandatory Z3 SMT verification significantly improves the pipeline's reliability for complex specifications by ensuring logical integrity. Input abstraction also enhances efficiency and consistency by standardizing inputs and reducing tokens. However, limitations persist: the simplification step can fail Z3 verification for exceptionally long statements, and the final assessment of the natural language output's quality and semantic accuracy still requires manual human evaluation.

4. EVALUATION AND LIMITATIONS

This section evaluates our automation pipeline's (Section 3) performance on 31 complex logical specifications previously identified (Section 2) as resistant to direct LLM translation. out of 74 specifications, 31 proved resistant to clear and accurate direct LLM translation due to complexities like deep nesting, negations, or excessive length. These 31 intractable cases formed the testbed for our pipeline. Our pipeline, which incorporates crucial Z3 SMT verification for logical equivalence, successfully processed 27 of these 31 inputs (87.10%, Table 1). The four SMT processing failures, attributed to exceptionally lengthy statements (over 10 distinct variables) where Z3 verification likely did not succeed, were consequently deemed automatic failures for the natural language (NL) quality assessment, as no trustworthy translation could be evaluated. NL outputs from the 27 SMT-verified specifications then underwent rigorous human evaluation by one of the authors. A detailed rubric guided this assessment of Readability, Understandability, Accuracy (defined as faithfully representing the SMT-verified logic in clear NL), and Conciseness, aiming for objective judgment despite inherent NL assessment challenges. An output "Passed" only if achieving 'Good' ratings (≥4 on a 5-point scale) across all criteria. While evaluating natural language can be subjective, we contend that the accuracy of a translation against its source logical specification remains verifiable through careful human evaluation; the four instances that failed the overall quality assessment correspond precisely to those that did not pass the pipeline's internal Z3 SMT verification stage.

Evaluation Outcome	Count	Rate	Interpretation
Passed All Criteria	27/31	87.10%	High quality on all criteria
Failed All Criteria	4/31	12.90%	Poor quality on all criteria

Table 1: Automation Evaluation

This non-generation of a verified output led directly to their automatic 'Failed' score in the quality assessment, reinforcing that successful Z3 verification is an essential prerequisite for our pipeline to produce usable natural language.

5. DISCUSSION

Direct LLM translation struggles with complex formal specifications. Our verified automation pipeline, employing SMT verification and simplification before LLM translation, effectively overcomes this, processing ~87% of previously intractable cases and yielding high-quality outputs (~84% rated 'Good' overall). This validates combining formal methods (for logical integrity) with LLMs (for translating verified, simpler logic). The primary limitation is handling exceptionally long statements (~13% failure rate). Future work should target this length issue and automated NL quality evaluation **[4].** Our hybrid approach significantly advances reliable specification translation for human understanding.

Acknowledgements: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(RS-2021-NR060080).

References: [1] Natasha Yogananda Jeppu, Tom Melham, and Daniel Kroening. 2023. Enhancing active model learning with equivalence checking using simulation relations. Form. Methods Syst. Des. 61, 2–3 (Aug. 2023), 164–197**.**
[2] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824-24837.
[3] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. arXiv preprint arXiv:2305.10601
[4] Mendoza, K., Trippel, C., et al. (2024). Translating Natural Language to Temporal Logics with Large Language Models and Model Checkers. Proceedings of the Formal Methods in Computer-Aided Design (FMCAD).
[5] Salama, M., El-Gazzar, R., & Nagi, M. (2024). Using Large Language Models for Natural Language Processing in Requirements Engineering: A Systematic Guideline. Requirements Engineering.