In AAAI 2024 Spring Symposium on User-Aligned Assessment of Adaptive AI Systems (AIA 2024)
Stanford, CA, USA

Can LLMs translate SATisfactorily? Assessing LLMs in Generating and Interpreting Formal Specifications

Rushang Karia, Daksh Dobhal, Daniel Bramblett, Pulkit Verma, and **Siddharth Srivastava**
School of Computing and Augmented Intelligence
Arizona State University
Tempe, Arizona 85281 USA
{rushang.karia, ddobhal, drbrambl, verma.pulkit, siddharths}@asu.edu

Abstract

Stakeholders often describe system requirements using natural language which are then converted to formal syntax by a domain-expert leading to increased design costs. This paper assesses the capabilities of Large Language Models (LLMs) in converting between natural language descriptions and formal specifications. Existing work has evaluated the capabilities of LLMs in generating formal syntax such as source code but such experiments are typically hand-crafted and use problems that are likely to be in the training set of LLMs, and often require human-annotated datasets. We propose an approach that can use two copies of an LLM in conjunction with an off-the-shelf SAT solver to automatically evaluate its translation abilities without any additional human input. Our approach generates formal syntax in the form of SAT formulae to automatically generate a dataset. We conduct an empirical evaluation to measure the accuracy of this translation task and show that SOTA LLMs cannot adequately solve this task, limiting their current utility in the design of complex systems.

1 Introduction

Automatic system synthesis and verification often require specifications to be provided in a formal language such as propositional logic (Haubelt and Feldmann 2003; Scholl and Becker 2001). Typically, human experts serve as middlemen that can (a) translate natural language (NL) specifications of stakeholders to formal syntax, or (b) explain or interpret the system's functionality by translating the system manual into NL. Given the success of Large Language Models (LLMs) in translation tasks (Xue et al. 2021), utilizing LLMs as middlemen can help in reducing overall system design costs. Thus, it is vital to develop an evaluation methodology that can assess the capabilities of LLMs in such settings.

However, developing such a methodology is quite difficult. Firstly, obtaining high-quality datasets - such as those that contain ground truth data that LLMs have not been trained on - is difficult. As LLMs evolve, the dataset would need to evolve as well since it would likely be included as a part of the next-gen LLMs training process. Scaling up existing datasets is challenging since they require human annotators to encode NL text and their formal specifications. Finally, the assessment task must consider both the directions of translation; formal-to-natural and natural-to-formal. Existing approaches for evaluating LLMs often lack in one of these dimensions.

Our Contributions We present a scalable approach for assessing LLMs w.r.t. their capabilities in translating formal syntax in a handsfree fashion. Our key contributions are:

1. Inspired by real-world system specifications, we propose the generation of boolean satisfiability (SAT) based datasets that can be generated randomly using generators and can be categorized by complexity.
2. We propose an automatic, handsfree approach that allows the bidirectional assessment of the translation task using two copies of an LLM by using off-the-shelf SAT solvers to evaluate the translation accuracy.
3. We motivate research in this area by conducting an empirical evaluation and showcasing that current SOTA LLMs are lacking even on simple formal specifications.

There has been plenty of work on SAT reasoning using LLMs (Fan et al. 2023; Pan et al. 2023; Tian et al. 2021). These methods are orthogonal to the translation task considered in this paper which relates to generating descriptions of SAT formulae and converting NL text to formulae.

Shi et al. (2022) use minimum Bayes risk decoding to translate natural language to formal syntax like source code. Their methodology uses existing datasets for assessing the accuracy and cannot convert code to natural language. StarCoder (Li et al. 2023) is a language model that can generate as well as summarise code. One key disadvantage of this approach is the reliance on expert, human-annotated datasets in their evaluation.

In contrast to existing literature, our approach can accurately assess an LLM without requiring human intervention and can automatically scale datasets.

2 Assessing LLM SAT Translation

We now provide some background followed by a description of our approach.

Formal Framework We consider formal specifications that are expressed as boolean satisfiability (B-SAT) formulae using propositional logic (Biere et al. 2021). Given a set of n boolean variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and boolean operators representing negation ($\neg$), disjunction ($\vee$), and con-

AAAI 2024 Spring Symposium on User-Aligned Assessment of Adaptive AI Systems. Stanford University, Stanford, CA, USA.

junction ($\wedge$), a SAT formula f is obtained by recursively applying the grammar $\mathcal{G}_{sat} \rightarrow x|p \vee p'|p \wedge p'|\neg p$ where $x \in \mathcal{X}$. The canonical representation of formulae obtained using $\mathcal{G}_{sat}$ is the conjunctive normal form (CNF). A formula f is in $(k, m)-$CNF form if $f \equiv f_1 \wedge \ldots \wedge f_m$ where $f_i = p_1 \vee \ldots \vee p_k$ and $p_i = \{x_j, \neg x_j\}$ with $x_j \in \mathcal{X}$. Given an assignment X of truth values to every variable in $\mathcal{X}$, $f(X)$ is the truth value of the formula. Two formulae f_1, f_2 are equivalent $f_1 \equiv f_2$ if they have the same truth value for all possible assignments using $\mathcal{X}$, i.e. $\forall X f_1(X) = f_2(X)$.

The LLM SAT translation task (NL↔SAT) Given an LLM, the *SAT→NL* translation task involves converting a SAT formula f to an NL description. Similarly, *NL→SAT* translates an NL description to a SAT formula.

Our Approach Let e be a non-deterministic function that translates a natural language string s to a SAT formula f. Similarly, let d be a non-deterministic function that translates f to s. e and d thus serve as an encoder and decoder that can perform *SAT→NL* and *NL→SAT* respectively. In general, there are many possible correct encodings $e(f)$ and decodings $d(s)$ for a given f and s. Thus, the functions e^{-1} and d^{-1} are not well-defined.

Our key observation is that if e and d come from the same system (e.g. a neural network or LLM), then we can check the accuracy of the system by composing e and d. Let f be a SAT formula. Now, if the system preserves truth in both translations, then $d(s)$ will be a factual representation of f and $f' = d(e(f))$ will be equivalent to f even if f and f' are not syntactically identical. Since $e(f)$ produces natural language description it is quite challenging to check whether $e(f)$ is a factual representation of f without human intervention. However, we can use off-the-shelf SAT solvers like Z3 (de Moura and Bjørner 2008) to check if $f \equiv d(e(f))$.

We use the above insights to automatically assess the *SAT→NL* and *NL→SAT* capabilities of LLMs (i.e. e and d are represented by the same LLM). Since LLMs utilize context windows to change their output, we use two different copies of the same LLM so that there is no contextual knowledge being exchanged between the encode-decode process.

Dataset Generation We create high-quality datasets by using SAT formulae generators that use $\mathcal{G}_{sat}$ to generate formulae. One key benefit of using such generators is that can they can generate SAT formulae with a certain structure or complexity class. For example, $(k, m)-$CNF generators generate structured CNF formulae and it is well-known that for a formula with n variables, there is a ratio $r = m/n$ where the difficulty of the problem increases (Selman, Mitchell, and Levesque 1996).

3 Empirical Evaluation

We used GPT-4 (OpenAI 2023b), GPT-3.5-turbo (OpenAI 2023a), Mistral-7B-Instruct (Mistral AI 2023), and Gemini Pro (Google 2023) as the SOTA LLMs in our evaluation. We evaluate whether they are effective for *NL↔SAT*. As a result, we used a simple setting $k = n = 3$ in creating our dataset. Real-world systems use values for k, n that are much higher.

We tested our prompts by generating a dataset $\mathcal{D}_{cnf}$ of 400 different $(k, m = rn)-$CNF formulae by varying r from 1.0 to 7.0 using a step size of 0.5. CNF formulae are structured and easy to describe making them a good benchmark for testing the efficacy of our prompts. The *SAT→NL* prompt asks an LLM to convert a SAT formula to an NL description whereas the *NL→SAT* prompt asks to convert the NL description to a SAT formula and output only the SAT formula with no other text. We iteratively modified our prompts until at least one LLM (GPT-4 in our case) was able to achieve $\geq 95\%$ accuracy in *NL↔SAT* on $\mathcal{D}_{cnf}$.

CNF formulae serve as a good test bed for prompts but human stakeholders are unlikely to understand or describe system capabilities in such a format. Thus, our evaluation tests the efficacy of SOTA LLMs on *NL↔SAT* using randomly generated formulae. To do this, we randomly generated 400 different formulae by recursively applying $\mathcal{G}_{sat}$. For a CNF formula in $\mathcal{D}_{cnf}$ with ratio r, we generated a comparable random formula such that the total number of operators ($\wedge$, $\vee$) are equal in both.

We used the same prompt for all LLMs. Finally, we used a temperature of 0.1 for all models.

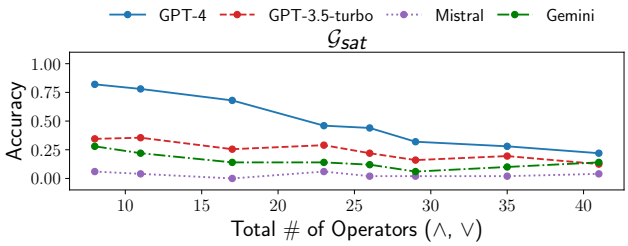

Figure 1: Accuracies (higher values better) of various SOTA LLMs on *NL↔SAT* on randomly generated formulae.

Results Our results are presented in Fig.1. It is clear from our results that current SOTA LLMs are not performant in the *NL↔SAT* task. As the size of the formula (the total number of conjunctions and disjunctions) increases, the performance degrades across all LLMs. These results are even surprising for GPT-4 whose accuracy on comparable CNF formulae was always $\geq 95\%$. We describe some of the errors that cause the low accuracy of the LLMs.

SAT→NL Errors: One of the most common errors in this translation was messing the order of the parentheses. The LLMs were not able to effectively describe the formulae taking into account the parentheses.

NL→SAT Errors: Hallucinations and negating propositions were common errors even when the NL sentence was sufficient for a human to correctly decode it to a SAT formula.

4 Conclusions and Future Work

We develop an approach that allows for effective assessment in the formal translation capabilities of SOTA LLMs. Our approach does not require human annotations to verify the accuracy of translation. Our results show that there is much to be done before LLMs can be deployed in translating formal syntax. We plan to investigate approaches that can help improve performance in future work.

Acknowledgements

The project was supported in part by the Arizona State University's GPSA Jumpstart Research Grant.

References

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*. IOS Press. ISBN 978-1-64368-160-3.

de Moura, L. M.; and Bjørner, N. S. 2008. Z3: An Efficient SMT Solver. In *Proc. TACAS*.

Fan, L.; Hua, W.; Li, L.; Ling, H.; Zhang, Y.; and Hemphill, L. 2023. NPHardEval: Dynamic Benchmark on Reasoning Ability of Large Language Models via Complexity Classes. arXiv:2312.14890.

Google. 2023. Gemini Pro. https://arxiv.org/pdf/2312.11805.pdf. Accessed: 2023-01-10.

Haubelt, C.; and Feldmann, R. 2003. SAT-based Techniques in System Synthesis. In *Proc DATE*.

Li, R.; Allal, L. B.; Zi, Y.; Muennighoff, N.; Kocetkov, D.; Mou, C.; Marone, M.; Akiki, C.; Li, J.; Chim, J.; Liu, Q.; Zheltonozhskii, E.; Zhuo, T. Y.; Wang, T.; Dehaene, O.; Davaadorj, M.; Lamy-Poirier, J.; Monteiro, J.; Shliazhko, O.; Gontier, N.; Meade, N.; Zebaze, A.; Yee, M.; Umapathi, L. K.; Zhu, J.; Lipkin, B.; Oblokulov, M.; Wang, Z.; V, R. M.; Stillerman, J.; Patel, S. S.; Abulkhanov, D.; Zocca, M.; Dey, M.; Zhang, Z.; Moustafa-Fahmy, N.; Bhattacharyya, U.; Yu, W.; Singh, S.; Luccioni, S.; Villegas, P.; Kunakov, M.; Zhdanov, F.; Romero, M.; Lee, T.; Timor, N.; Ding, J.; Schlesinger, C.; Schoelkopf, H.; Ebert, J.; Dao, T.; Mishra, M.; Gu, A.; Robinson, J.; Anderson, C. J.; Dolan-Gavitt, B.; Contractor, D.; Reddy, S.; Fried, D.; Bahdanau, D.; Jernite, Y.; Ferrandis, C. M.; Hughes, S.; Wolf, T.; Guha, A.; von Werra, L.; and de Vries, H. 2023. StarCoder: May the Source be With You! *Transactions on Machine Learning Research*.

Mistral AI. 2023. Mistral-7B Instruct v0.2. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2. Accessed: 2023-01-10.

OpenAI. 2023a. GPT-3.5-turbo-0613. https://platform.openai.com/docs/models/gpt-3-5. Accessed: 2023-01-10.

OpenAI. 2023b. GPT-4-1106-preview. https://arxiv.org/pdf/2303.08774.pdf. Accessed: 2023-01-10.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. Y. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In *Proc. EMNLP Findings*.

Scholl, C.; and Becker, B. 2001. Checking Equivalence for Partial Implementations. In *Proc. DAC*.

Selman, B.; Mitchell, D. G.; and Levesque, H. J. 1996. Generating Hard Satisfiability Problems. *AIJ*, 81(1-2): 17–29.

Shi, F.; Fried, D.; Ghazvininejad, M.; Zettlemoyer, L.; and Wang, S. I. 2022. Natural Language to Code Translation with Execution. In *Proc. EMNLP*.

Tian, J.; Li, Y.; Chen, W.; Xiao, L.; He, H.; and Jin, Y. 2021. Diagnosing the First-Order Logical Reasoning Ability Through LogicNLI. In *Proc. EMNLP*.

Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al-Rfou, R.; Siddhant, A.; Barua, A.; and Raffel, C. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proc. NAACL-HLT*.