Dehallucinating Large Language Models Using Formal Methods Guided Iterative Prompting

Susmit Jha
Computer Science Laboratory
SRI International
Menlo Park, USA
susmit.jha@sri.com

Sumit Jha
Department of Computer Science
University of Texas, San Antonio
San Antonio, USA
sumit.jha@utsa.edu

Patrick Lincoln
Computer Science Laboratory
SRI International
Menlo Park, USA
patrick.lincoln@sri.com

Nathaniel D. Bastian
Army Cyber Institute
United States Military Academy
West Point, New York, USA
nathaniel.bastian@westpoint.edu

Alvaro Velasquez
dept. name of
name of organization
City, Country
email address or ORCID

Sandeep Neema
dept. name of
name of organization
City, Country
email address or ORCID

***Abstract*—The large language models (LLMs) such as ChatGPT have been trained to generate human-like responses to natural language prompts. LLMs use a vast corpus of text data for training, and can generate coherent and contextually relevant responses to a wide range of questions and statements. Despite this remarkable progress, LLMs are prone to *hallucinations* making their application to safety-critical applications such as autonomous systems difficult. The hallucinations in LLMs refer to instances where the model generates responses that are not factually accurate or contextually appropriate. These hallucinations can occur due to a variety of factors, such as the model's lack of real-world knowledge, the influence of biased or inaccurate training data, or the model's tendency to generate responses based on statistical patterns rather than a true understanding of the input. While these hallucinations are problematic in tasks such as text summarization and question-answering, they can be catastrophic when LLMs are used in autonomy-relevant applications such as planning. In this work-in-progress paper, we focus on the application of LLM in autonomous systems and sketch a novel self-monitoring and iterative prompting architecture that uses formal methods to automatically detect these errors in the LLM response. We exploit the dialog capability of LLMs such as GPT to iteratively steer them to responses that are consistent with our correctness specification. We report preliminary experiments with ChatGPT that show the promise of the proposed approach on tasks such as automated planning.**

I. Introduction

Large language models [1]–[4] have revolutionized natural language processing tasks, such as text generation, translation, and text summarization. The training data used for LLMs such as ChatGPT are massive corpora of text documents from the internet, including books, news, science articles, public code repositories, and websites. The training data is preprocessed to remove non-textual elements and then fed into a transformer neural network architecture [5] that is trained as an unsupervised auto-regressive generative model. The large size (GPT-3 [3] used over 45 terabytes of data) and diversity of the training data allow ChatGPT to generate a wide range of responses to natural language prompts.

However, these models are prone to generating hallucinations [6]–[9], that is, outputs that are factually incorrect or contextually inappropriate. The hallucination problem is a critical challenge that must be addressed to ensure the reliability and trustworthiness of large language models. The encoding of knowledge in LLMs is lossy and the knowledge generalization naturally leads to memory distortion and inaccurate retrieval of knowledge from even training data. Thus, this problem cannot be resolved by simply scaling data and models. Moreover, recent studies have shown that large language models can also suffer from the stale information problem, where the model's outputs are based on outdated or inaccurate knowledge due to the dataset's bias or the model's inability to keep up with the evolving knowledge. In this paper, we investigate the problem of hallucination and factually incorrect responses from LLMs and describe how formal methods can help alleviate this problem in the context of autonomy.

Formal methods has been extensively used in cyber-physical and autonomous systems to provide rigorous guarantees on the behavior of a system and to verify whether it meets the expected specifications. The limited scalability of the underlying combinatorial search such as satisfiability solving [10], mixed-integer linear programming [11], and logic programming [12] in the formal verification methods has impeded their wide-scale adoption. We build on the recent demonstrations [13], [14] that LLMs can be used as an efficient but not trustworthy inductive search engine in autonomy-relevant tasks such as planning. We posit that this trust deficit in LLMs because of the hallucination phenomenon can be addressed using formal verification to detect the inconsistencies and iteratively prompt the LLM using the dialog capability till it converges to a correct response acceptable to the formal verifier.

This is a first step towards integrating deductive formal methods with inductive large language models into a high-assurance learning and reasoning architecture.

II. Related Work

A number of techniques have been explored to improve the accuracy and the reliability of the large language models, such as training the model on more diverse and representative data, and integrating external sources of knowledge into the model's responses. These techniques broadly fall into the following categories:

1) Fine-tuning: Fine-tuning [15] can reduce hallucination in LLMs by allowing them to adapt to specific tasks or domains, and to learn from more targeted and focused training data. But fine-tuning is computationally and financially expensive for the LLMs with hundreds of billions of parameters. For many domains, it is not obvious what is a sufficient size of domain-specific dataset that needs to be used for fine-tuning, and continuously monitoring and fine-tuning LLMs is not practical. Failure of fine-tuning to improve LLM performance has been reported in literature [16]. Further, fine-tuning [17] is known to adversely impact the model's fluency, conversational capability, and in-context learning ability [18] which is critical to its response to prompts.
2) Knowledge graphs: Another approach is to connect LLMs to knowledge graphs [19], which represent knowledge as a graph of interconnected entities and relationships. Knowledge graphs can be used to encode a wide range of structured and unstructured knowledge, including facts, concepts, and relationships, and can provide a rich source of information for LLMs to draw upon. Methods have been developed to infuse structured knowledge into LLMs by directly training models on factual triples of knowledge graphs (KGs) and such models pre-trained on knowledge graphs have been shown to outperform baselines [20]. But this requires a well-curated and complete knowledge base, building which is a time-consuming and expensive endeavor. Maintaining these knowledge bases with consistent and up-to-date information over time is also challenging.
3) Memory augmentation: External knowledge can be encoded into a key-value memory that exploits the fast maximum inner product search for memory querying. These memory slots can then be integrated with language models [21] for relatively smaller models, such as T5 [22]. Such a memory augmentation has been shown to improve the performance of the deep learning model on knowledge-intensive tasks. This method has not been yet attempted on large language models such as GPT3 and ChatGPT. Hence, the impact of these approaches on the other characteristics of the LLMs such as fluency and in-context learning is not known.

In autonomy applications, the space of possible problems is very large and can have many factors of syntactic variation which may not even be relevant to the underlying search problem. It has been recently shown that LLMs are very sensitive to such irrelevant variations [23]. Consequently, improving accuracy via expensive fine-tuning or explicit curation of knowledge graphs in such a context would be unrealistic. Further, we avoid any change in the network architecture such as memory augmentation, and hence, our approach can be deployed on any state-of-the-art conversational LLM even though our current experiments are only with ChatGPT.

III. Technical Approach

We use the LLM as an oracle and our approach relies on a formal approach to prompt engineering for detecting and removing hallucination bugs. This loose coupling with LLM enables our approach to be agnostic to the specific conversational LLM.

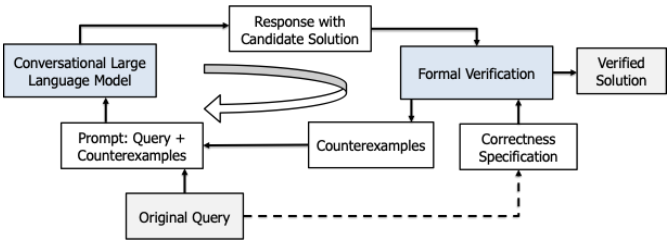

Fig. 1. The proposed architecture for combining formal methods and the large language model. The prompt to the large language model is constructed by concatenating the original query with feedback from the formal verification engine identifying the incorrect candidate solutions as counterexamples (with optional explanations that can be produced by verifiers). The correctness specification is constructed from the query and is currently not automated. The counterexample-guided iterative refinement of prompts ends when the LLM produces a solution that the verifier finds to be correct.

Figure 1 describes the architecture of how a formal verifier can be used to check the output of the LLMs and in the event of incorrect responses, these are detected as counterexamples and added as a part of the prompt to the LLM. This concatenation can be explicit addition to the prompt or just addition to the conversation when the LLM is conversational and capable of a dialog such as ChatGPT. Formal verification engines also have the capability of constructing small counterexamples that detect the part of the solution that is incorrect. This is helpful in detecting with fine granularity the hallucinated fact by the LLM and the revised prompt that eliminate this hallucination. Iteratively, we keep expanding the prompt of the LLM to eliminates the counterexamples (with optional explanations localizing the relevant part of the counterexample). The LLMs do not have any guarantees on learning monotonically or even being consistent with the prompt, and hence, the iterations are not guaranteed to terminate even when the number of possible counterexamples is finite. But in practice, our preliminary experiments indicate the LLMs eliminate the identified hallucinations and converge to a response that is verified to be correct.

The simple architecture described in Figure 1 is effective in combining the LLMs and formal verifiers because of the following empirical observations:

- LLMs can be prompted to generate well-structured outputs that can be easily parsed by a formal verification engine. The inclusion of a large corpus of code in their training data can explain this observation. We successfully use this to ensure the output of LLMs can be ingested by the verifier.

We have not currently used LLMs to automatically derive the correctness specification, but it is likely this can be successful for a wide class of specifications.

- Formal verifiers are very fast at checking the correctness of a single solution for a task such as planning (as compared to solving the problem using combinatorial search). Further, verification can be used to detect the part of the solution that violates the specification and to provide a more informative feedback to the LLMs that rule out not just a single hallucinated solution but a whole class of incorrect solutions. Our preliminary investigation has used a rudimentary verifier and fault localizer, but there exists a huge body of work on verification, fault isolation and explanation generation that can be leveraged using this architecture.
- LLMs demonstrate remarkable in-context learning [18] and adding counterexamples and explanations to the prompt steers them away from incorrect responses and effectively dehallucinates them, into eventually producing a correct solution that is accepted by the verifier. The large context length used to train LLMs appears to be responsible for their ability to stay consistent with provided prompts. As the number of counterexamples increase for more complex problems and the length of the prompt becomes longer, it would be interesting to observe whether LLMs still remain consistent with the provided prompts and the feedback provided by the formal verifiers.
- The use of formal methods for tasks such as planning requires tedious modeling into existing solvers and verifiers. Further, the responses of these formal engines need to be translated into more easily interpretable form to be used by engineers and developers who are not experts in formal methods. The use of LLM in the above architecture serves as a human-friendly frontend and, thus, partially alleviates the challenge of making formal methods more accessible.

For application of this method to automated planning, we use a simple satisfiability solver based plan verifier where we encode the specification of the final goal and each action manually (automated generation of this specification from the natural language text is feasible using fine-tuned LLMs and is left as a future exercise). Now, given a sequence of actions constituting the plan identified by the LLM, we check whether the sequence satisfies the specification. This is a much simpler query than finding a plan that satisfies the specification (which would require solving a quantified satisfiability problem). We append all the failed plans as "not valid" to the prompt and query the LLM to produce new plans. We terminate when the LLM produces a valid plan.

IV. Experiments

Our experiments comprise two case-studies, and we use ChatGPT as the LLM. The first case study is a planning problem with three blocks that has been widely studied in literature. The second case study is a robot task planning problem.

In the first case-study (see Figure 2), we specify the high-level predicates that capture the state of the world, such as `BlockOn, OnTable,` and `Holding`. We also describe the permitted operations/actions and the initial state. We provide a final state and ask the LLM (ChatGPT) to produce a valid plan. ChatGPT takes 3 iterations to produce a correct plan. The verifier appends the incorrect sequences to the prompt only stating that these sequences are not valid. The final response from the LLM provides a human-readable justification in addition to the final plan.

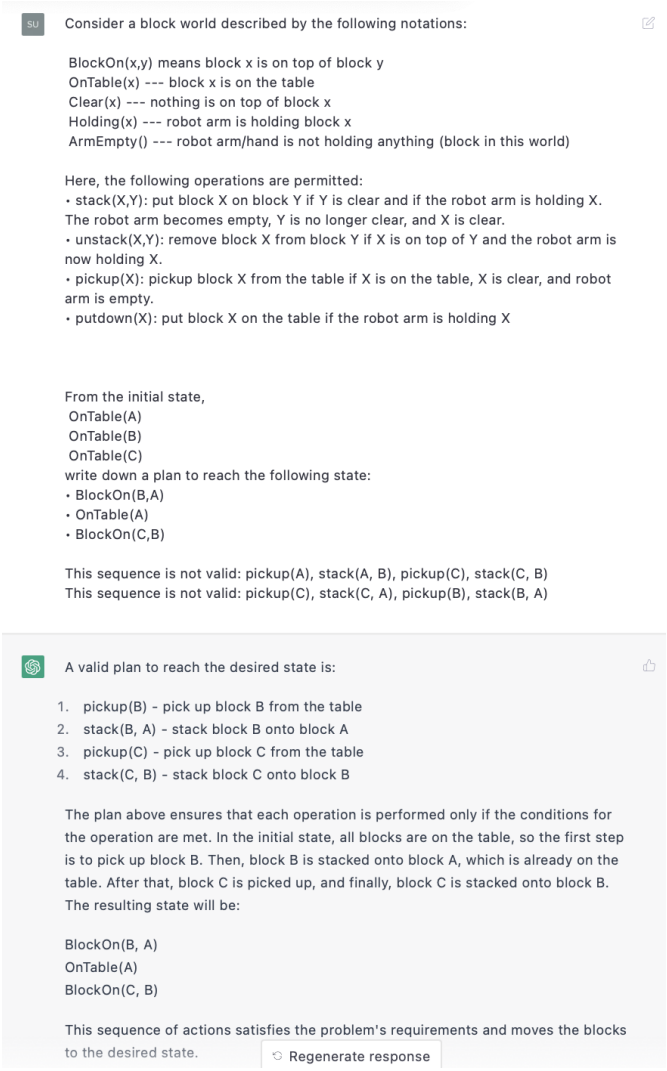

Fig. 2. ChatGPT Screenshot: The final prompt that includes (a) the initial query for the planning problem corresponding to stacking blocks on a table and (b) the added corrections after formal verification stating that the earlier suggested responses had invalid sequences. The final response is at the bottom, where ChatGPT comes up with the correct plan.

In the second case study (see Figure 3), we consider a house with 6 rooms with a particular connection topology. We ask the LLM to plan the movement of a robot (roomba) to clean all rooms and return to the bathroom using a set of primitives defined as part of the prompt. In this case, we provide feedback on the failed plans by specifying the reason for failure. The LLM converges again to a plan that is correct and avoids the invalid subsequences identified by the verifier.

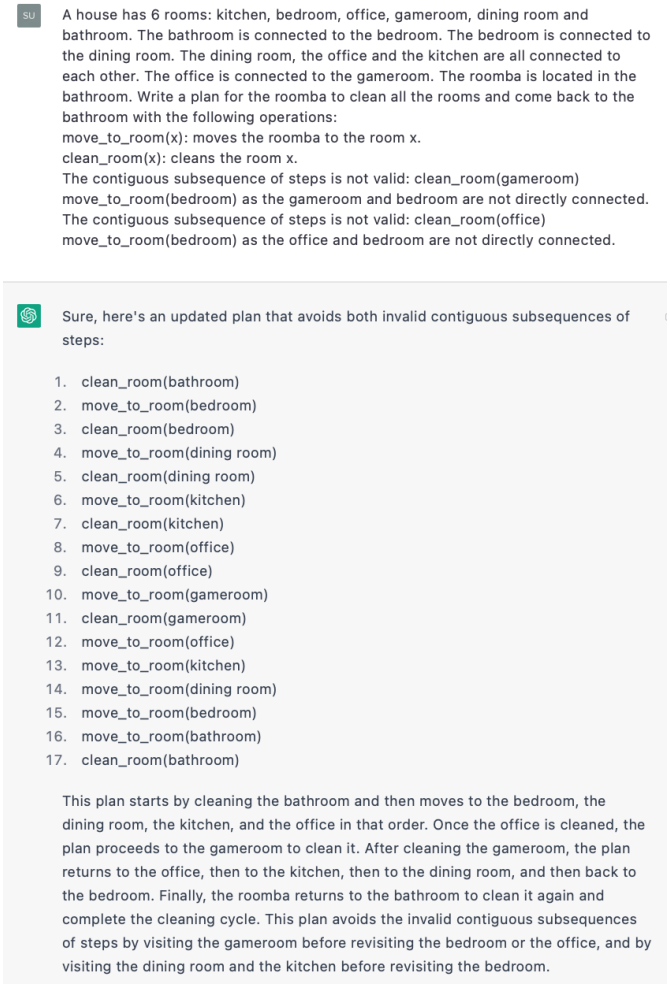

Fig. 3. ChatGPT Screenshot: The final prompt that includes (a) the initial query for the robot's task planning problem and (b) the added corrections after formal verification, stating that the earlier suggested responses had invalid sequences. In this case, we provide an additional explanation for why a sequence was invalid. The response is at the bottom where ChatGPT comes up with the correct plan.

V. Conclusion

Our approach to addressing the hallucination challenge can be viewed as an adversarial variant of the in-context learning approach, wherein we use formal verification to detect incorrect responses and include that as a part of the prompt in the dialog with the LLM. Our method does require formal modeling to check the responses of the LLM, but we do not need to formally model the LLM itself. The initial experiments reported over the planning task in this work-in-progress paper are encouraging and indicate that the proposed combination of LLMs and formal verifiers can alleviate the problem of hallucination in LLMs that is essential for safety-critical applications. In ongoing work, we are investigating methods to automatically extract the specification to be used by a formal verifier from the prompt text. We are also exploring the use of more advanced formal verification techniques, such as those based on model checking temporal properties.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

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

[4] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res*, vol. 23, pp. 1–40, 2021.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.

[7] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, *et al.*, "Recipes for building an open-domain chatbot," *arXiv preprint arXiv:2004.13637*, 2020.

[8] M. Cao, Y. Dong, and J. C. K. Cheung, "Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization," in *Proceedings of the ACL (Volume 1: Long Papers)*, pp. 3340–3354, 2022.

[9] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, 2022.

[10] M. Y. Vardi, "Boolean satisfiability: theory and engineering," *Communications of the ACM*, vol. 57, no. 3, pp. 5–5, 2014.

[11] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, pp. 449–481, 2013.

[12] J. W. Lloyd, *Foundations of logic programming*. Springer Science & Business Media, 2012.

[13] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[14] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*, pp. 9118–9147, PMLR, 2022.

[15] C. Lee, K. Cho, and W. Kang, "Mixout: Effective regularization to finetune large-scale pretrained language models," *arXiv preprint arXiv:1909.11299*, 2019.

[16] M. Bommarito II and D. M. Katz, "Gpt takes the bar exam," *arXiv preprint arXiv:2212.14402*, 2022.

[17] Y. Wang, S. Si, D. Li, M. Lukasik, F. Yu, C.-J. Hsieh, I. S. Dhillon, and S. Kumar, "Preserving in-context learning ability in large language model fine-tuning," *arXiv preprint arXiv:2211.00635*, 2022.

[18] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?," *arXiv preprint arXiv:2202.12837*, 2022.

[19] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[20] F. Moiseev, Z. Dong, E. Alfonseca, and M. Jaggi, "Skill: Structured knowledge infusion for large language models," *arXiv preprint arXiv:2205.08184*, 2022.

[21] Y. Wu, Y. Zhao, B. Hu, P. Minervini, P. Stenetorp, and S. Riedel, "An efficient memory-augmented transformer for knowledge-intensive nlp tasks," *arXiv preprint arXiv:2210.16773*, 2022.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[23] S. K. Jha, R. Ewetz, A. Velasquez, and S. Jha, "Responsible reasoning with large language models and the impact of proper nouns," in *Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022*.