Harnessing LLMs for Verification-Guided Specification Repair

Alberto Tagliaferro[0009−0004−4923−7831]

alberto.tagliaferro@polimi.it
Politecnico di Milano, Milan, 20133, Italy

Abstract. The increasing complexity of software-intensive systems necessitates efficient and user-friendly methods for specifying multi-agent missions. This work envisions a framework exploiting generative AI to generate formally-verified specifications based on a Domain-Specific Language (DSL). The generated DSL specification is automatically translated into a selected formal model. The formal model undergoes verification, yielding quality metrics regarding the specification and its correctness. Results are examined to provide repair feedback actions, enhancing the initial prompt for better performance.

Keywords: Multi-agent Systems · Domain-Specific Languages · Large Language Models · Iterative Specification Repair

1 Motivations and Goals

The ever-growing complexity of software-intensive systems, such as Internet of Things (IoT), smart manufacturing, healthcare, and assistive robotics, underscores the increasing need of solutions for specifying interactive multi-agent system applications quickly and unambiguously, even for non-expert users. In addition to ensuring the accessibility of specifications for software-intensive systems—e.g., System-of-Systems (SoSs)—it is crucial to guarantee systems' reliability, especially given their critical nature and operation in uncertain environments. This requirement underscores the necessity for the proposed solution to facilitate user-friendly specification and ensure the reliability and robustness of SoSs [2].

A common way to address these issues is the adoption (or the development from scratch) of a Domain-Specific Language (DSL), providing a high level of abstraction [1]. In a model-driven approach, a DSL specification can be compiled to generate a formal model (e.g., based on automata or logics) or directly create the code for the agents. In both cases, the artifacts generated are often complex and challenging for users to specify directly. The obtained artifacts must then be verified (e.g., through model checking or other techniques for program verification) to compute quality attributes and assess their correctness.

However, existing approaches find limited applicability due to the complexity of software-intensive systems. Specifically, we identify two pressing challenges:

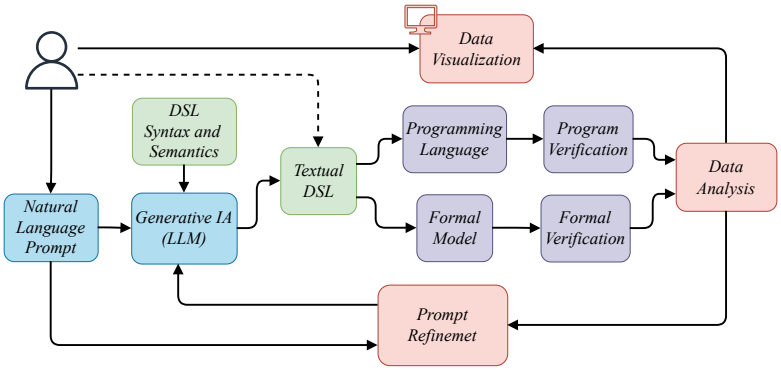

Fig. 1: Envisioned framework. The color-coding is light blue boxes for the generative AI elements, the green elements for the DSL-based blocks, purple for formal components, and red for the new main component of this proposal, used for the visualization and refinement of the outcome.

I. Translating requirements expressed in natural language into a DSL can prove challenging, time-consuming, and error-prone for users. Moreover, this process demands that the user have a thorough understanding of language syntax and semantics when using high-level DSLs [3].
II. Understanding the outcomes of the formal analysis can be hard to interpret for humans [14].

This doctoral research proposal aims to address and investigate possible solutions to these challenges, exploiting the combination of formal verification and cutting-edge technologies such as generative AI and data analysis.

2 Envisioned Framework

The framework illustrated in Figure 1 outlines the key components necessary to address the described issues, starting from a specification expressed with *Natural Language Prompt* and incorporating a *Prompt Refinement* mechanism.

The entry point to the approach is the user prompting the *Generative AI*, describing in natural language the agents' goal and the operational environment. The *Generative AI*, possibly taking as input also the foundation of the target *DSL Syntax and Semantics*, provides as output the *Textual DSL* specification for the mission defined by the user. The generation of the *Textual DSL* instead of the target formal model offers two significant advantages. Firstly, the generated outcomes are easier for the human user to modify once it is generated, as they are more comprehensible compared to the final specification (i.e., *Formal Model* or *Programming Language*). Secondly, Large Language Models (LLMs) have been specifically developed for text generation, making a DSL a more suitable and

natural output compared to a mathematical formulation. The AI-generated DSL specification is then translated into the *Formal Model* (resp., *Programming Language*) and subject to *Formal* (resp., *Program*) *Verification*. The used verification technique depends on the nature of the artifacts created from the DSL.

The outcome of the verification might be hard to interpret for the user, especially concerning how to revise the generated DSL in light of such results. The *Data Analysis* in Figure 1, combined with the *Data Visualization* element, facilitates human interpretation. The analyzer yields the specification of repair action to be forwarded to the *Prompt Refiner*. The latter, starting from the initial *Natural Language Prompt*, implements the repair action and updates the prompt for the *Generative AI* (LLM), aiming to obtain a revised specification. This iterative process can be repeated multiple times to meet the user's requirements.

A potential validation test for the envisioned framework is that of LIrAs, a domain-agnostic DSL for the specification of multi-agent systems [12]. The DSL is then translated into Stochastic Hybrid Automata (SHA) [5], exploiting expert-defined formalizations of the agents' primitive skill. However, depending on the specific agent's capabilities, the latter cannot be inferred directly from the DSL. This challenge can be addressed by exploiting automata learning [8]. The iterative refinement approach shown in Figure 1, combined with automata learning, has the potential to significantly reduce the end-user effort in specifying and verifying multi-agent missions.

3 Related Works

This section reports an analysis of the state of the art concerning the potential of generative AI in specifying agents' tasks, focussing on the robotic field.

Stella et al. [11] analyze the potential changes that LLMs could bring in the coming years, particularly in simplifying robotic design (e.g., GPT is an LLM also utilized for robotics applications [13]). Singh et al. [10] introduce ProgPrompt, an LLM prompt structure designed to generate plans across various situated environments, robot capabilities, and tasks. Notably, fed with a prompt describing the goal and including primitives, objects, and example tasks, ProgPrompt generates a plan based on a specification. Similarly, Liang et al. [6] present an approach to exploring LLMs to generate code for robotic specifications. CodeBolter and RoboEval [4] represent robot-agnostic tools for programming services from natural language inputs and evaluating them. Luan et al. [7] demonstrate how developing robot specifications can be enhanced by utilizing multiple LLMs rather than relying on a single one. Finally, Ross et al. [9] demonstrate a different application of LLMs, focusing on program generation to assist software development based on natural language inputs.

This brief analysis of LLMs for robot specifications underpins the need for a structured and sufficiently flexible framework that combines LLMs and formal verification with custom DSLs (unseen during LLM training). Such a framework should also support sufficiently complex agent-based applications, including multi-agent and human-agent systems.

4 Research Plan Status

This project is currently at the conceptual stage. The immediate focus is conducting a comprehensive literature review (briefly summarized in Section 3) to consolidate the identified challenges. Concurrently, generative AI development and data analysis techniques will be evaluated to select the most appropriate tools for developing and testing the proposed solutions. These foundational steps are crucial for establishing a solid basis for the project's subsequent phases, which will involve designing, implementing, and validating the presented toolchain.

References

1. Dragule, S., Gonzalo, S.G., Berger, T., Pelliccione, P.: Languages for specifying missions of robotic applications. Software Engineering for Robotics pp. 377–411 (2021)
2. Ferreira, F.H., Nakagawa, E.Y., dos Santos, R.P.: Reliability in software-intensive systems: challenges, solutions, and future perspectives. In: Euromicro Conference on Software Engineering and Advanced Applications. pp. 54–61. IEEE (2021)
3. Gray, J., Fisher, K., Consel, C., Karsai, G., Mernik, M., Tolvanen, J.P.: DSLs: the good, the bad, and the ugly. In: ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications. pp. 791–794 (2008)
4. Hu, Z., Lucchetti, F., Schlesinger, C., Saxena, Y., Freeman, A., Modak, S., Guha, A., Biswas, J.: Deploying and evaluating LLMs to program service mobile robots. IEEE Robotics and Automation Letters (2024)
5. Lestingi, L., Zerla, D., Bersani, M.M., Rossi, M.: Specification, stochastic modeling and analysis of interactive service robotic applications. Robotics and Autonomous Systems **163**, 104387 (2023)
6. Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., Zeng, A.: Code as policies: Language model programs for embodied control. In: IEEE Intl. Conf. on Robotics and Automation. pp. 9493–9500. IEEE (2023)
7. Luan, Z., Lai, Y.: Automatic robotic development through collaborative framework by large language models. arXiv preprint arXiv:2402.03699 (2024)
8. Narendra, K.S., Thathachar, M.A.: Learning automata: an introduction. Courier corporation (2012)
9. Ross, S.I., Martinez, F., Houde, S., Muller, M., Weisz, J.D.: The programmer's assistant: Conversational interaction with a large language model for software development. In: Intl. Conf. on Intelligent User Interfaces. pp. 491–514 (2023)
10. Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., Garg, A.: ProgPrompt: program generation for situated robot task planning using large language models. Autonomous Robots **47**(8), 999–1012 (2023)
11. Stella, F., Della Santina, C., Hughes, J.: How can LLMs transform the robotic design process? Nature Machine Intelligence **5**(6), 561–564 (2023)
12. Tagliaferro, A., Lestingi, L., Rossi, M.: Towards verifiable multi-agent interaction pattern specification. In: IEEE/ACM Intl. Conf. on Formal Methods in Software Engineering. pp. 122–126 (2024)
13. Vemprala, S., Bonatti, R., Bucker, A., Kapoor, A.: ChatGPT for robotics: Design principles and model abilities. arXiv preprint arXiv:2306.17582 (2023)
14. Zimmerman, M.K., Lundqvist, K., Leveson, N.: Investigating the readability of state-based formal requirements specification languages. In: Intl. Conf. on Software engineering. pp. 33–43 (2002)