UNIVERSITÀ POLITECNICA DELLE MARCHE
Repository ISTITUZIONALE

Bridging Large Language Models and Logic Programming

This is the peer reviewd version of the followng article:

Original
Bridging Large Language Models and Logic Programming / Crocetti, V.; Dragoni, A. F.. - (2025), pp. 1066-1071. (4th IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering, MetroXRAINE 2025 Ancona, IT 22 - 24 October 2025) [10.1109/MetroXRAINE66377.2025.11340308].

Availability:
This version is available at: 11566/355412 since: 2026-04-09T19:51:03Z

Publisher:
Institute of Electrical and Electronics Engineers Inc.

Published
DOI:10.1109/MetroXRAINE66377.2025.11340308

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. The use of copyrighted works requires the consent of the rights' holder (author or publisher). Works made available under a Creative Commons license or a Publisher's custom-made license can be used according to the terms and conditions contained therein. See editor's website for further information and terms and conditions.
This item was downloaded from IRIS Università Politecnica delle Marche (https://iris.univpm.it). When citing, please refer to the published version.

Publisher copyright:

IEEE - Postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. To access the final edited and published work see 10.1109/MetroXRAINE66377.2025.11340308

(Article begins on next page)

23 April 2026

Bridging Large Language Models and Logic Programming

1nd Valerio Crocetti
Dipartimento Ingegneria dell'Informazione
Università Politecnica delle Marche
UNIVPM
60131 Ancona, Italy

2nd Aldo Franco Dragoni
Dipartimento Ingegneria dell'Informazione
Università Politecnica delle Marche
UNIVPM
60131 Ancona, Italy

***Abstract*—This paper proposes a new framework, connecting natural language and logic programming languages, through which the automatic generation of executable Prolog code from queries of the user expressed in natural language. Exploiting Large Language Models (LLMs), the framework transforms the input problems in natural language into Prolog syntax, runs the implemented logic using a middleware layer connected to the SWI-Prolog engine, and performs a reverse transformation into natural language. Unlike general-purpose systems that integrate LLMs with symbolic solvers for diverse reasoning tasks, this work focuses on leveraging Prolog to enhance reasoning reliability and ensure verifiable outputs.**

Experimental comparisons in fact show that the framework can outperform the base LLMs on select symbolic reasoning tasks. These findings are obtained without the need for Prolog-specific extension of the LLM, which demonstrates the utility of the framework to enhance reasoning accuracy when the base model fails to produce logical valid code. Although the base LLM is performing best on average, the framework does show a clear advantage in tasks demanding both accuracy and verifiable answers.

For demonstration of its practical value, the framework has been deployed within a chat interface, which illustrates how it could make logic programming accessible to non-experts. Yet, the modularity and adaptability of the framework go well beyond this particular application and form a base from which to better LLM performance via symbolic reasoning and to develop at the same time avenues for using natural language understanding with formal logics systems.

***Index Terms*—LLM, Prolog, Qwen, chatbot, AI, logic programming**

I. Extended Abstract

Logic programming[1], specifically Prolog [**?**], is perhaps the most widely recognized symbolic language and an effective paradigm for knowledge representation and problem solving in situations that require symbolic computation, logical reasoning, and the modeling of complex dependencies. Nevertheless, the technical syntax and declarative paradigm of Prolog is very difficult to be widely used. This paper introduces a framework that leverages Large Language Models (LLMs) to bridge the gap between natural language and Prolog, enabling users to harness the benefits of logic programming without prior expertise in its syntax.

The framework converts natural language problem statements into Prolog code via a pre-trained LLM (i.e., Qwen [2] implementation). The workflow consists, first, in the creation of Prolog code guided by a well-tailored prompt, then in the cleaning of the provided code to extract the most useful logical clauses and queries, and finally in the running of this cleaned code with SWI-Prolog[3]. The resulting output is reverse-translated into natural language for user interpretation, with the original Prolog code made available for further study and learning.

One of the main innovation of the framework is the iterative error-handling scheme. If execution errors happen, the system generates refined code by asking the LLM for rephrased problem description on the basis of error message from Prolog and the original description. This iterative process, which may include adjustments to parameters like LLM temperature to encourage diverse solutions, continues until successful execution or a retry limit is reached. This iterative update not only increases the stability of code generation, but also demonstrates the ability of the framework to expand reasoning through symbolic logic.

Although previous studies have addressed LLM applications in logic programming, e.g., code completion, logical form translation, and neuro-symbolic reasoning, this framework combines those tasks into an end-to-end LLM system with a deep focus on automated error-correction and refinement. As a result of its modular architecture, the framework has a potential generalizability to situations that are not merely question answering, and has potential for more general applications, where symbolic thinking and strict logical precision may be desired.

As evidence of its practical value, the framework has been realized as part of a chatbot interface, demonstrating how Prolog-based reasoning can be made available to users in the world, i.e. this application illustrates the broader promise for enhancing LLM reliability and verifiability of the framework by combining natural language understanding with formal logic. By lowering the entry barrier to logic programming and expanding its applications, this work contributes to advancing the integration of symbolic reasoning into modern AI systems. The combination of Large Language Models (LLMs) and logic programming has increasingly been a lively field of research, including numerous studies that address different challenges in this integration, such as code generation or symbolic reasoning. One such contribution is made by Shin et al. [4],

who explored the translation of natural language into formal logic. Their contribution dictates the translation component of this framework, in which natural languages syntaxes map onto formal logical syntaxes. Nevertheless, the paper method further generalizes this idea by fusing the translation in a more concrete way into Prolog code generation and execution. This allows this framework to not only to natural language to logic, but also to execute and validate generated Prolog code, thereby providing us with a more complete solution for logical reasoning.

In program synthesis, effort, e.g., LogiGLUE[5], is focused on automating logical reasoning, and although this approach also targets the automation of logical reasoning, it is different from other work in that it implicitly embeds an error-driven refinement scheme. The generator repeatedly optimizes the created code with respect to the occurrence of execution errors and, in conjunction with feedback from the Prolog engine, ensures an updated output. By iteratively refining the reasoning accuracy, in particular very importantly for LLMs where the impact of a single error in reasoning can be massively important in the output, this process is demonstrated to improve the reasoning accuracy.

Neuro-symbolic systems, as in NS-CL [6], can offer intriguing perspectives, integrating neural and symbolic AI for the solution of such difficult tasks. In contrast, as NS-CL is concerned with general concept learning in different domains, this work is more specific. Automated generation of Prolog code, however, specifically targeted toward logic programming tasks was the goal. The framework allows best of all worlds, not to say that it is a solution automation workflow, but writing code, executing code, error correction of code.

Moreover, studies such as that of Yang et al [**?**] explored the task of generating Prolog code from formula, offering a more domain-specific implementation of LLMs in logic programming. This framework, however, is designed for broader applicability. This paper deals with a major class of logic programming problems, and introduces a natural language interface that can be used by any persons who have no knowledge on these topics. Together with the iterative error refinement procedure this interface guarantees that the generated code, not only functionally correct but also logically correct.

A recent contribution by Pan et al. (2023) [7] further explores LLM integration with logic programming. Their research offers an analogous iterative error-refinement loop, such that the logically generated output can be refined iteratively, or cyclically. In this framework, we also take into account the dynamic updates of the temperature parameter, which in this work has a key role to play in the variety and quality of the generated solutions. As the temperature is gradually increased, the proposed framework pushes the LLM to search around a wider search space for solutions, which finally results in a more robust and generalizable reasoning process.

This paper presents an end-to-end general solution that integrates LLMs and Prolog on one hand, on the other to code generation and code execution. Compared with other approaches which are usually used to narrow down logic programming, or are not sufficiently mature to carry out robust error handling, the system treats such aspects as a core issue and allows automatic error correction, dynamic temperature adaptation, and ordered code refinement. Not only does this approach ensure the correctness of the output, but it also gives the system the ability to solve a more general class of logical problems more easily, i.e., the logical reasoning based on LLM is further accessible and believable to a broader audience.

Evaluations conducted on the Qwen2.5-Coder-32B-Instruct and Qwen2.5-72B-Instruct models demonstrate the effectiveness of the proposed framework on the complex logical reasoning task with high accuracy. While the chatbot application is used as a practical demonstration of the capabilities of the framework, its usability extends beyond this application, showing the wider applicability of the system to a variety of logic programming scenarios.

Future work will consider extending the framework to address a larger class of problem domains and, also, to cover more sophisticated Prolog extensions. Furthermore, more explorations will be performed on the integration of multiple LLMs and other approaches to provide code generation with better quality and stability. This framework is a highly relevant advance in bringing logic programming to a general audience, as well as exploiting the capabilities of LLMs in performing complex reasoning tasks, which opens new perspectives in both research and application domains.

REFERENCES

[1] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed. Cambridge University Press, 2017.

[2] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," Alibaba Cloud, Tech. Rep., 2024.

[3] J. Wielemaker *et al.*, "SWI-Prolog," https://www.swi-prolog.org/, Accessed: 2024-01-10.

[4] R. Shin, C. Lin, S. Thomson, C. Chen, S. Roy, E. A. Platanios, A. Pauls, D. Klein, J. Eisner, and B. Van Durme, "Constrained language models yield few-shot semantic parsers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7699–7715. [Online]. Available: https://aclanthology.org/2021.emnlp-main.608/

[5] M. Luo, S. Kumbhar, M. shen, M. Parmar, N. Varshney, P. Banerjee, S. Aditya, and C. Baral, "Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models," 2024, unpublished. [Online]. Available: https://arxiv.org/abs/2310.00836

[6] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," in *International Conference on Learning Representations*, 2019.

[7] L. Pan, A. Albalak, X. Wang, and W. Wang, "Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3806–3824. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.248/