ChatIDP: An Interactive chatbot for IDP Knowledge Bases

Benjamin Callewaert[1,2,3] and Joost Vennekens[1,2,3]

[1] KU Leuven, De Nayer Campus, Sint Katelijne Waver, Belgium,
[2] Leuven.AI, Dept. of Computer Science,
[3] Flanders Make – DTAI-FET

Abstract. ChatIDP is a chatbot that simplifies interacting with the IDP reasoning engine by allowing natural language questions. It combines language model capabilities with IDP's logical reasoning for accurate, domain-specific answers.

Keywords: Knowledge-Based System · Chatbot · IDP

1 Introduction

ChatIDP is a chatbot developed with two primary goals. First, it serves as a user-friendly interface for the IDP reasoning engine, developed at KU Leuven. Unlike other interfaces, such as the Interactive Consultant or traditional API, which require users to understand the logical inference mechanisms of the IDP system, ChatIDP offers a more accessible alternative. By allowing users to pose questions in natural language, ChatIDP makes the IDP engine available to a broader audience.

Second, ChatIDP can serve as a neuro-symbolic framework that combines the conversational abilities of language models with the logical reasoning strengths of IDP. This integration allows the chatbot to provide accurate and truthful answers within specific domains, enhancing its reliability and broadening its applications.

2 Preliminaries

The **IDP-Z3 system** [6] is a reasoning engine for FO($\cdot$), a rich extension of First-Order Logic (FOL). It implements the philosophy of the Knowledge Base Paradigm [7]: knowledge is represented in a purely declarative manner, independent from how it is used. This is done by storing the knowledge in a Knowledge Base (KB), to which then various inference tasks can be applied to put it to practical use.

The KB itself consists of three types of blocks: A *vocabulary* specifies a set of *type*, *predicate*, or *function* symbols. A *structure* provides an interpretation for the symbols in its vocabulary. A *theory* contains a set of logical formulas, written in FO($\cdot$). By itself, the KB is not executable: it merely represents the knowledge of a domain. To put this knowledge to use, the IDP system offers multiple inference tasks, like propagation and model expansion. The **Interactive**

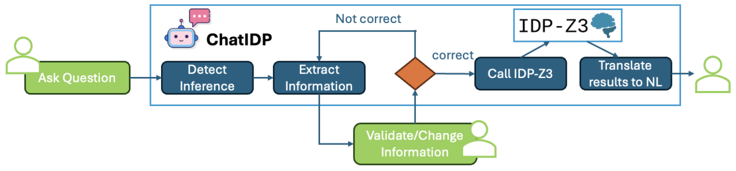

Fig. 1. Overview of steps executed by ChatIDP

Consultant [4] is a generic, user-friendly, interactive interface for the IDP-Z3 system. The interface displays a *tile* for each symbol in the KB, which allows users to assign values to these symbols. In the past, the IDP system has already proven itself as a suitable tool for applications in diverse domains [2,3,9,10]. The latest version of the IDP system, used in this work, is IDP-Z3 [5].

3 Design

The design of ChatIDP involves a sequence of steps, as depicted in Figure 1, to ensure accurate and user-friendly interaction with the IDP reasoning engine. The user first uploads the relevant IDP KB and asks a question. ChatIDP then identifies the appropriate inference and extracts the relevant information, which the user can validate before the IDP engine is called. Finally, the results of the call are translated back into NL and presented to the user.

Inference Detection is crucial for identifying the correct reasoning task IDP-Z3 should perform. The possible inferences include: (1) *Model Expansion*: Generate solutions for the problem; (2) *Model Check*: Verify if a solution exists; (3) *Optimization*: Find the optimal model for a specific symbol; (4) *Propagation*: Derive all logical conclusions; and (5) *Explain*: Explain why a symbol has a specific value or why no solutions exist. A BERT sequence classification model[8], trained on a dataset of natural language questions, detects the intended inference from the user's question.

The next step is **information extraction**, where a small language model (Microsoft's Phi-3-mini SML [1]) uses few-shot prompting to identify and translate key details from the user's question into a partial interpretation of the symbols in the IDP program. For now, we only focussed on extracting information for propositional logic programs.

This structured process enables ChatIDP to efficiently interface with the IDP system, ensuring that user questions are accurately interpreted and addressed.

4 Conclusion & Future Work

We developed chatIDP, a chatbot that provides an intuitive natural language interface for interacting with an IDP KB, eliminating the need to understand logical inference, and making it more accessible than existing IDP interfaces like the Interactive Consultant and traditional API. In the future, we will focus on extending chatIDP beyond propositional logic and testing its performance in answering domain-specific questions against standard LLMs and other neuro-symbolic methods.

References

1. Abdin, M., Jacobs, S.A., Awan, A.A., Aneja, J., Awadallah, A., Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Behl, H., et al.: Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219 (2024)
2. Aerts, B., Deryck, M., Vennekens, J.: Knowledge-based decision support for machine component design: A case study. Expert Systems with Applications **187**, 115869 (Jan 2022). https://doi.org/10.1016/j.eswa.2021.115869
3. Carbonnelle, P., Bogaerts, B., Vennekens, J., Denecker, M.: Interactive Configuration Problems in Observable Environments p. 8 (2022)
4. Carbonnelle, P., Deryck, M., Vennekens, J., et al.: An interactive consultant. In: BNAIC, Date: 2019/11/06-2019/11/08, Location: Bruxelles (2019)
5. Carbonnelle, P., Vandevelde, S., Vennekens, J., Denecker, M.: Interactive configurator with fo(.) and idp-z3 (2023)
6. De Cat, B., Bogaerts, B., Bruynooghe, M., Janssens, G., Denecker, M.: Predicate logic as a modeling language: the IDP system. In: Kifer, M., Liu, Y.A. (eds.) Declarative Logic Programming: Theory, Systems, and Applications, pp. 279–323. ACM (Sep 2018). https://doi.org/10.1145/3191315.3191321, https://dl.acm.org/citation.cfm?id=3191321
7. Denecker, M., Vennekens, J.: Building a Knowledge Base System for an Integration of Logic Programming and Classical Logic. In: Garcia de la Banda, M., Pontelli, E. (eds.) Logic Programming, vol. 5366, pp. 71–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89982-2_12, http://link.springer.com/10.1007/978-3-540-89982-2_12, series Title: Lecture Notes in Computer Science
8. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
9. Van Hertum, P., Dasseville, I., Janssens, G., Denecker, M.: The KB paradigm and its application to interactive configuration. Theory and Practice of Logic Programming **17**(1), 91–117 (2017)
10. Vlaeminck, H., Vennekens, J., Denecker, M.: A logical framework for configuration software. In: Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming. p. 141–148. PPDP '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1599410.1599428, https://doi.org/10.1145/1599410.1599428