Teaching Prolog and Logic Programming with Jupyter Notebooks

Galileo Sartor[1,*], Adam Wyner[1]

[1]Swansea University

Abstract

This short paper presents the use of a JupyterHub server to teach Prolog and Logic Programming in the context of a University course in Artificial Intelligence. The server is used to simplify how student exercises are distributed, collected, accessed, and automatically assessed. The server allows the opportunity to provide a standardized environment for students to test their code. With the right mix of addons and extensions, it can be used to explore graphical web applications built with Prolog as the backend. The paper discusses the context of the course, the implementation of the server, and how it was used to teach Prolog and Logic Programming in an interactive way.

Keywords

Prolog, Logic Programming, Jupyter, Artificial Intelligence, Education,

1. Introduction

This paper presents the use of a JupyterHub server for Prolog programming in the context of a University course in Artificial Intelligence.

Students are familiar with online tools and with interactive programming systems, so when they encounter a Prolog prompt for the first time they may find it challenging. In fact, it may be that they are not familiar with this style of interaction with computers, especially at the beginning of their academic life. It is also worth asking what the purpose of a course in Artificial Intelligence is, and in particular how to focus on the understanding of Logic programming and its role, rather than specific technical setups and interactions.

Additionally, by leveraging the Jupyter framework it is possible to show the students how they can visualize what they develop in a more natural way. Linking Prolog, web technologies, and visualisations can also be used to develop simple, accessible programs such as games with a Prolog backend.

2. Prior Work

There is a substantial body of literature on teaching Prolog and logic programming, reflecting decades of experience in both computer science and artificial intelligence curricula. Early work by Clocksin and Mellish [1] established Prolog as a language suitable for introducing declarative programming concepts. Prolog has been used to teach logic programming principles, problem-solving techniques, and the foundations of artificial intelligence [2, 3]. Subsequent studies have explored pedagogical strategies, such as using visualizations [2], interactive environments [4], and problem-based learning [5] to enhance student engagement and comprehension.

Several textbooks and course designs emphasize the importance of relating logic programming to familiar computational paradigms, while also highlighting its unique features, such as unification and backtracking [3]. More recent approaches leverage other systems to build interactive programs based on Prolog [6], using web-based tools and notebooks to lower the barrier to entry and provide immediate

Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming, September 9–13, 2025, Rende, Italy

*Corresponding author.

galileo.sartor@swansea.ac.uk (G. Sartor); a.z.wyner@swansea.ac.uk (A. Wyner)

feedback and test-driven learning. Examples of this are the Ciao Prolog Playground [4], SWISH [7], and GUPU [8]. Focusing more specifically on teaching AI, there are several online resources [9, 10, 11].

Despite these advances, challenges remain in helping students to add declarative thinking to their computational thinking as an addition to imperative programming. In addition, there are challenges in designing assessments that accurately capture such conceptual understanding [5].

Overall, the literature suggests that effective teaching of Prolog benefits from a combination of interactive tools, carefully chosen examples, and explicit connections to broader themes in computer science and artificial intelligence.

In the world of Generative AI, it is important to teach Prolog in a way that highlights its strengths in reasoning and knowledge representation, especially as these concepts become increasingly relevant in AI applications in different domains such as the law [12] and the arts [13]. It is also worthwhile considering its potential interaction with the new generative systems [14, 15].

Recent work has highlighted the importance of integrating logic programming with contemporary AI topics, including explainable AI and knowledge representation [15, 7]. These connections help students appreciate the relevance of logic programming in modern AI systems and foster a deeper understanding of foundational concepts.

3. Context

The course in which we decided to embed and test the JupyterHub Server is a course in Artificial Intelligence, in the second year of the Bachelor in Computer Science at Swansea University. In the course, the students are introduced to the basic concepts of symbolic AI, with a focus on Logic Programming. The rationale is that sub-symbolic approaches (machine learning) and data science are taught elsewhere in the curriculum, while the mathematical underpinnings, particularly logic, are largely not operationalised. In our view, logic is an essential component of computational thinking. Moreover, the knowledge and skills taught under the rubric of symbolic AI such as search methods, knowledge representation and reasoning, syntactic and semantic parsing, planning, and causal reasoning have been and will continue to be invaluable. This is particularly so where one is an analyst, designer, and evaluator with respect to some new problem. It is preferable to have a diversity of tools in one's computational toolbox, which are appropriate to the problem at hand and integrated with other tools for an overall solution. In this, teaching symbolic AI using Prolog contributes to so called *neuro-symbolic AI.*

Most students have already been exposed previously to a short introduction to declarative programming (including Prolog) in a prior module. The course is delivered with both frontal lectures and weekly lab sessions, where students can practice the concepts introduced in the lectures with a set of tasks and exercises provided.

The initial use case of the JupyterHub server was to simplify the distribution and collection of the exercises, giving students a simple way to access their work and submit it with continuous automated tests to help them solve the exercises. The server also proved useful for students who wished to use their own computer, as it allowed them to access the provided notebooks and exercises as well as their previous work, while maintaining a standard environment. The programming environment does not require installing anything locally, so it is easier to onboard students focusing less on installing on different systems and configurations.

The server is based on the work done at the University of Düsseldorf to build a Jupyter Kernel for Prolog[1], used for the course in ProB[2]. The version used in the course is based on SWI-Prolog, with a server based installation using the compact Littlest JupyterHub[3]. The server is hosted by Swansea University on a virtual machine with sufficient resources to support the entire classes simultaneously.

[1]Available at https://github.com/hhu-stups/prolog-jupyter-kernel
[2]Available at https://gitlab.cs.uni-duesseldorf.de/general/stups/prob-teaching-notebooks/-/tree/master/
[3]Available at https://tljh.jupyter.org/en/latest

4. Implementation and Use

As mentioned in section 3, the JupyterHub server implementation is based on an existing Prolog Jupyter Kernel. Changes were made to the server to enable the use of nbgrader[4], a Jupyter extension that allows instructors to create assignments which, once completed and submitted by the students, are automatically graded. Automated tests are embedded within the assignments to provide immediate feedback to students. When evaluating the student submissions there is also a custom grading option to allow for qualitative assessment on the style/functionality of the code. The automated tests give students immediate feedback on their work, allowing them to iterate and improve their solutions before the final submission.

Having set up the server, the next step was to create a set of notebooks that could be used in the course. The notebooks were designed to cover the main topics of the course, including an introduction to Prolog, to more advanced topics such as DCGs, constraint logic programming, and meta-programming (e.g., building an expert system shell to query the user).

Each notebook contains a mix of explanations, examples, and exercises. While this collection was created independently, a translator script was developed to convert to/from SWISH notebooks, and a similar tool was developed for Ciao Prolog Active Logic Documents [4]. Technically notebooks are a mixture of Prolog and Markdown text (as their Python counterparts), allowing students to read explanations and examples, and then to write their own code in the same environment. The notebooks can also include both visible tests that can be run to check the correctness of the code written by the students, providing immediate feedback on their work, as well as hidden ones that are used for evaluation after submission.

4.1. Server specific configuration

The server authentication was integrated with the university's single sign-on system, allowing students to log in using their institutional credentials. This streamlined the onboarding process and reduced administrative overhead.

SWI-Prolog was chosen as the Prolog implementation, as it is used in a course most students took previously and was supported already by the Prolog Jupyter kernel. The Kernel itself is designed to permit expansion with other implementations if desired.

The server environment can be customized with pre-installed libraries, such as support for sCASP [4] and Logical English [16] along with tools to enhance the Prolog programming experience like Prolog Syntax highlighting.

Overall, the implementation focused on minimizing technical barriers for students, enabling them to concentrate on learning Prolog and logic programming concepts. The combination of JupyterHub, the Prolog kernel, and nbgrader created an interactive and supportive environment for both teaching and learning.

The adoption of JupyterHub with a Prolog kernel in the context of teaching logic programming has yielded several notable outcomes and insights. The interactive nature of Jupyter notebooks significantly lowers the entry barrier for students, especially those less familiar with command-line interfaces or traditional Prolog environments. The ability to interleave explanatory text, code, and immediate feedback within a single document can foster a more engaging and iterative learning process, where the students can assess their understanding and if needed roll back to previous tasks. At the moment there was no user study carried out, so this result is an impression from the lectures and lab sessions, as well as discussions with students.

Automated assessment through nbgrader proved useful for both students and instructors. Students benefited from instant feedback on their solutions, which encouraged experimentation and self-directed learning. Instructors, on the other hand, were able to efficiently manage assignments, monitor student progress, and focus their attention on conceptual misunderstandings rather than routine grading.

[4]Available at https://nbgrader.readthedocs.io/en/stable/

4.2. Web Integration

Part of this work was to integrate the Kernel with technologies the students were already familiar with and to show how Prolog can be used as the engine for graphical applications, such as simple games. For this purpose we created simple web applications that use Prolog as the backend. The first demo is a simple Sudoku game, where one can play the game in a web browser. In this case Prolog is used to check the validity of the user input or to solve the game. This was built with the possibility to show or edit the Prolog code, so students could see what was driving the game validation and try changing the rules to implement different variations of the game.

Another goal with this sample web application was to show students how they could distribute small Prolog programs, as they can integrate Prolog with other technologies they might be more familiar with (HTML, CSS), and distribute a complete package, with graphical capabilities and no complex installation required.

While this portion of the course was not further developed, it may be a way to get students interested in writing a small game or application themselves in the context of a course project. The idea is to show how Prolog can be used to build applications that are not just about knowledge representation, but also about building interactive systems that can be used in a web browser.

4.3. Critical thinking in the age of Generative AI

One of the driving forces behind this work is the need to teach students how to think critically about the different tools they use, and how to pick the right one for the task at hand. With the pervasive exposure of students to Generative AI, it is important to understand what the strengths and weaknesses of these systems are, and how they can be used to complement each other. Not only is the teaching of Prolog and Logic Programming important in knowledge representation and reasoning, or to understand the fundamental importance of explanations and justifications in AI, but it is also important to understand what the different systems are useful for in simple practical scenarios.

Consider the following contrast. Suppose a student could use (or train) a Generative AI system to solve Sudoku puzzles. But, would this be a good use of the tools available? How would it contribute to the student's intellectual and skills development, which may be called upon in future to solve problems? In our point of view it is important to understand that for a similar application, where you can express the rules of the game, and where you can check the validity of a solution, it is better to use a Prolog program, as it will be more efficient, it will always give a valid solution (if one exists), and it will be easier to understand and explain. Sudoku is a simple example, but it can be extended to more complex problems, such as games with more complex rules, or different applications that require reasoning about the world in a clear and transparent way. The sudoku example is a useful demonstration, as there are many different rulesets that can be implemented making small changes to the Prolog rules.

5. Future Work

The JupyterHub server has been used successfully in the course, and it would be interesting to extend its use to other programming courses. Using Jupyter as a platform enables the use of other languages and tools, that can complement the Prolog development, building user interfaces, or interacting with other programming paradigms. Some of these capabilities are already available in Jupyter (Python) notebooks, and they would need to be tied in to Prolog, such as UI menus/fields like web forms, or graph generation, etc.

Another update would be to rebuild the Jupyter Kernel component to support the WebAssembly version of SWI-Prolog, which would allow running without a server, or using the server only for sharing notebooks and code. Finally, the system could be integrated with a comprehensive set of lectures and exercises, helping build shared course material for teaching Prolog and Logic Programming, which could be used by other universities and institutions.

Regarding the web integration, it would be interesting to extend the demos to include more complex applications. This could be done by building a set of demos that show how Prolog can be used to build interactive systems that can be used in a web browser, and how these systems can be used to solve real-world problems e.g., in the medical or legal domain.

Finally, to better assess the usefulness of such a system, it would be a good idea to carry out a proper user study, and enable a continuous feedback with students that can suggest changes or inform us on what works and what needs revision. This can also be done directly in Jupyter, to simplify the requirements and encourage feedback.

6. Conclusion

In this paper, we have described the deployment and use of a JupyterHub server with a Prolog kernel to support the teaching of Prolog and logic programming in a university setting. By leveraging Jupyter notebooks, nbgrader, and web integration, we have created an interactive and accessible environment that lowers technical barriers for students and enables immediate feedback and experimentation. This approach not only facilitates the learning of declarative programming concepts but also encourages students to critically evaluate the strengths of logic programming in comparison to contemporary generative AI systems. Our experience suggests that integrating modern tools and pedagogical strategies can enhance student engagement and understanding, and that further development of shared resources and interactive applications will continue to benefit both educators and learners in the field of artificial intelligence.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] W. F. Clocksin, C. S. Mellish, Programming in Prolog, Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55481-0.

[2] P. Brna, H. Pain, B. Du Boulay, Teaching, Learning and Using Prolog: Understanding Prolog 19 (1990) 247–256. URL: https://www.jstor.org/stable/23370479.

[3] L. Sterling, E. Y. Shapiro, The Art of Prolog: Advanced Programming Techniques, MIT Press, 1994.

[4] J. F. Morales, S. Abreu, D. Ferreiro, M. V. Hermenegildo, Teaching Prolog with Active Logic Documents, Springer Nature Switzerland, 2023, pp. 171–183. URL: https://doi.org/10.1007/978-3-031-35254-6_14. doi:10.1007/978-3-031-35254-6_14.

[5] H. Coelho, J. C. Cotta, Prolog by Example: How to Learn, Teach and Use It, Springer Science & Business Media, 2012.

[6] H. S. Pedro Ribeiro, Michel Ferreira, Teaching Artificial Intelligence and Logic Programming in a Competitive Environment 8 (2009) 85–100. URL: https://www.ceeol.com/search/article-detail?id=107606.

[7] J. Wielemaker, T. Lager, F. Riguzzi, SWISH: SWI-Prolog for Sharing, 2015. URL: http://arxiv.org/abs/1511.00915. doi:10.48550/arXiv.1511.00915. arXiv:1511.00915 [cs].

[8] U. Neumerkel, S. Kral, Declarative program development in Prolog with GUPU, 2002. URL: http://arxiv.org/abs/cs/0207044. doi:10.48550/arXiv.cs/0207044. arXiv:cs/0207044.

[9] D. L. Poole, A. K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2023. URL: https://www.cambridge.org/highereducation/books/artificial-intelligence/C113F6CE284AB00F5489EBA5A59B93B7. doi:10.1017/9781009258227.

[10] P. Blackburn, J. Bos, K. Striegnitz, Learn Prolog Now!, volume 7 of *Texts in Computing*, College Publications, 2006.

[11] M. Triska, The Power of Prolog: Introduction to modern Prolog, 2025. URL: https://github.com/triska/the-power-of-prolog.

[12] S. Sehgal, Y. A. Liu, Logical Lease Litigation: Prolog and LLMs for Rental Law Compliance in New York, Electronic Proceedings in Theoretical Computer Science 416 (2025) 59–68. URL: http://dx.doi.org/10.4204/EPTCS.416.4. doi:10.4204/eptcs.416.4.

[13] C. Jendreiko, Generative Logic: Teaching Prolog as Generative AI in Art and Design, volume 3799 of *CEUR Workshop Proceedings*, CEUR, 2024. URL: https://ceur-ws.org/Vol-3799/#paper9PEG2.0.

[14] P. Tarau, On Teaching Logic Programming in the Era of Generative AI, volume 3799 of *CEUR Workshop Proceedings*, CEUR, 2024. URL: https://ceur-ws.org/Vol-3799/#paper11PEG2.0.

[15] E. Saccon, A. Tikna, D. De Martini, E. Lamon, L. Palopoli, M. Roveri, When Prolog Meets Generative Models: a New Approach for Managing Knowledge and Planning in Robotic Applications, 2024, pp. 17065–17071. URL: https://ieeexplore.ieee.org/abstract/document/10610800. doi:10.1109/ICRA57147.2024.10610800.

[16] R. Kowalski, J. Dávila, G. Sartor, M. Calejo, Logical English for Law and Education, in: D. S. Warren, V. Dahl, T. Eiter, M. V. Hermenegildo, R. Kowalski, F. Rossi (Eds.), Prolog: The Next 50 Years, Lecture Notes in Computer Science, Springer Nature Switzerland, Cham, 2023, pp. 287–299. URL: https://doi.org/10.1007/978-3-031-35254-6_24. doi:10.1007/978-3-031-35254-6_24.