i-Prolog: A Web-Based Intelligent Tutoring System for Learning Prolog

Afroditi Stathaki[1], Haridimos Kondylakis[2(✉)], Emmanouil Marakakis[1], and Michael Kalogerakis[3]

[1] Department of Informatics Engineering, Technological Educational Institute of Crete, 71410 Heraklion, Greece
stathakiafrodith@hotmail.com, mmarak@cs.teicrete.gr
[2] Computational Biomedicine Laboratory, FORTH-ICS, N. Plastira 100, 70013 Heraklion, Greece
kondylak@ics.forth.gr
[3] Department of Electrical Engineering, Technological Educational Institute of Crete, 71410 Heraklion, Greece
mixalis@cs.teicrete.gr

Abstract. Intelligent tutoring systems (ITS) incorporate techniques for transferring knowledge and skills to students. These systems use a combination of computer-aided instruction methods and artificial intelligence. In this paper we present a web-based intelligent tutoring system. Although it can be used as a generic learning mechanism, in this paper, as a proof of concept we used it for learning Prolog. We present the architecture of our system and we provide details on each one of its modules. Each lesson includes the corresponding lecture with theory and exercises, a practice module where students can apply the corresponding theory and an assessment module to verify user's understanding. The system can be used with or without a teacher enabling distant learning. Among the novelties of our system is its flexibility to adapt to individual student choices and profile, offering a wide range of alternatives and trying to continuously keep the interest of the final user. The preliminary evaluation performed confirms the usability of our system and the benefits of using it for learning Prolog.

Keywords: Intelligent tutoring systems · Prolog

1 Introduction

Electronic tutors can deliver material through games, videos, movies, and exercises, a more exciting menu for any student and especially for those with learning difficulties. Electronic tutors try to personalize the learning process and to replace human tutors. The course content is personalized based on the evaluation of the student's performance and knowledge and as such the presentation of the material can differ for each student. In addition, computer tutors in many cases inflict less anxiety in students when compared to human tutors: Learners can practice as much as they want and make mistakes without anyone observing them.

A. L. Brooks et al. (Eds.): ArtsIT 2017/DLI 2017, LNICST 229, pp. 337–346, 2018.
https://doi.org/10.1007/978-3-319-76908-0_32

To this direction, many intelligent tutoring systems (ITS) have been constructed the last decades. According to Graesser [5], «ITSs are computerized learning environments that incorporate computational models from the cognitive sciences, learning sciences, computational linguistics, artificial intelligence, mathematics, and other fields». As such, ITSs have the common goal of enabling learning in a meaningful and effective manner by using a variety of computing technologies.

The primary goal of an ITS system is to mimic a human tutor by adapting its instructions according to individual student's performance, strengths and weaknesses [18]. There are many benefits associated with ITSs since they ease the work of instructors and can replace them in situations where they may not be present, such as with an embedded training system which may be used out in the field. Training is tailored and customized according to the potential of the individual student, enabling a more efficient student learning. Furthermore, the knowledge of the best instructors can be captured and incorporated into an ITS, and distributed to all students.

This paper focuses on the development of a web-based intelligent tutoring system. As a proof of concept we implemented the tool for Prolog, but it is built as generic as possible, able to be used for other lessons as well. Prolog is a logic programming language as its implementation is based on Horn Clause logic, a subset of first-order logic [19]. Prolog has built-in a unification algorithm and the only way to do a repetition is through recursion. As such many students coming from procedural programming find Prolog difficult to understand and use. To this direction, we designed a novel system enabling students to learn and experiment with Prolog. The designed system has many novel features and aims to teach Prolog programming and to be deployed in real-time education and/or in distance learning. The system presents the theory behind Prolog and examples and instructs the learners on how to construct Prolog programs. Then the practice module is used to enable practicing allowing students to use among three types of practicing: *hint-based*, *schema-based* or *open practice*. The whole learning environment is personalized according to individual profiles. It provides examples and appropriate instructions according to the personal learning style, making the whole learning process as personalized as possible. Then, an assessment module evaluates user understanding enabling or not the next lecture. A teacher is optional, and if present she/he can supervise the whole process asynchronously and identify the progress of the individual students.

The rest of this paper is structured as follows: In Sect. 2, we present related work. Then in Sect. 3 we present the architecture of our system and describe the components of our system. In Sect. 4, we present an initial evaluation of our system. Finally, Sect. 5 concludes this paper and presents directions for future work.

2 Related Work

There have been many ITSs developed over the last 20 years to help students learn geography, circuits, medical diagnosis [4], mathematics, physics, genetics, and chemistry.

For computer programming, ACT programming Tutor (APT) [2] offers hints for learning LISP, produced using a set of expert-defined production rules. Ask-Elle [3] is

a Haskell tutor that tries to help students by identifying program errors, whereas in ITAP [12] programming hints are employed using past student's data in order to debug Python programs. Those recent works show that hints are essential in tutoring systems. However, they mostly focus on debugging programs and not on combing theory teaching with program debugging.

For Prolog, there have been developed many university ITS for teaching programming, used with positive results [16, 17] and tools helping students to construct logic programs [10].

The Augmented Prolog Programming Environment (APPE) [11], for example, accepts student programs and detects errors from five categories: (1) Incorrect Solutions, (2) Uncovered Solutions, (3) Non termination, (4) Redundant Solutions, (5) Invalid Parameters. When an error is detected, advices are also provided to the students. Another ITS system for debugging Prolog programs by Looi [9], analyzes input programs and uses task-specific knowledge to detect bugs and to correct them. Belikova [1] on the other hand, separate the errors identified into two categories: The programs that cannot run because of errors and the ones that may work but possibly contain other errors. The work uses a set of recursive program schemata which cover several aspects of recursion such as simple recursion, monotonic or non-monotonic recursion etc. All these aforementioned works provide outdated tools and mostly focus on error-detection to guide Prolog learning. In our case however, the whole platform goes beyond simple debugging, provides lectures and example exercises and is personalized according to individual learning style. In addition, our system supports a more pedagogical learning approach and it is available for use on the web.

Another work closer to our vision is by Vlahavas et al. [15]. The system created can be used for distance learning and contains text, images and animated lessons. The system provides examples, bibliographic resources, exercises and many links for each lesson. Initially it provides the basic concepts to the amateur users and then progresses on information on more advanced topics. Finally, at the end, the user is guided in developing small Prolog programs. In our approach, we go beyond guided development of Prolog programs and we allow hint-based, schema-based and open practice. In addition, our system includes an assessment module for capturing user's progress and understanding.

Overall, although most of the developed systems can detect errors and some of them already provide useful resources to the end-users, still the intelligence used is limited. For example, the emotional status of the student could affect his/her learning curve and an ITS should be adaptable and personalized according to individual user profiles. In our approach however, user profiling is the central point of our implementation.

3 Architecture

The architecture of our system is shown in Fig. 1. It consists of six basic components: (1) The interface; (2) the tutoring/pedagogical module; (3) the teacher module; (4) the student/learner module; the (5) APIs for updating the Knowledge base and finally (6) the Knowledge Base. Bellow we provide more details for each one of these modules.

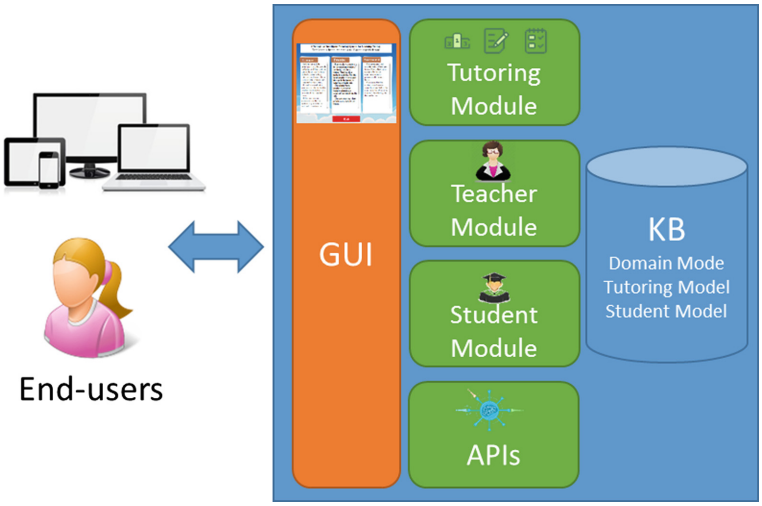

Fig. 1. The architecture of i-Prolog

We have to note that the system is developed both in the Greek and English language. The interface is developed using CSS, AJAX and HTML 5, whereas the back-end of the system is developed using Prolog.

3.1 The Tutoring/Pedagogical Module

This module in essence is the cognitive model that contains the concepts, the rules, and the problem-solving strategies of the domain to be learned. It controls the presentation of the material to be presented to the student and has a feedback mechanism in order to answer end-user's questions. It is responsible for selecting teaching goals, and for determining appropriate teaching strategies according to the selected student model, learner's needs and/or preferences, learning experiences, the domain of discourse and the instructional objectives of the intelligent tutoring system.

In our system, the information is structured as "lessons". For each lesson, we have the teaching goals and the teaching strategies. In addition, each lesson is divided into three parts: (a) the lecture (b) the practice and (c) the assessment part.

The *lecture part* supports several teaching methods, like question-answering, class style lecturing, etc. that can be combined, trying to keep alive the interest of the learner. In any case, the selection of the teaching method is guided by an underlying instructional theory, the theory of Skinner [14]. According to Skinner, a question is a stimulus and the answer is a reaction. If a student reacts, there is positive amplification and then a student can learn. As such, in general, a lecture first presents some theory items to the student, and then, in order to progress to the next theory item, he has to answer one or more small exercises/examples, based on the taught theory. To do that he can revisit the theory. However, she/he can progress to the next theory item, independent of the correctness of his/her answer.

The *practice part* on the other hand, includes different techniques for exercising. The system allows learners to discover which technique is more appropriate for them. The underling theories of this section are the theories of "*discovery learning*" and "*fact-finding learning*" claiming that learners must have a critical thinking and should discover by themselves what is important to learn [13]. As such, the learner can select according to his/her preference among *hint-based practice*, *schema-based practice* and *open-practice.* In the hint-based practice, many hints guide the learner to complete a certain task. In the schema-based approach example, schemata are suggested for completing the task showing examples and guiding the end-user on constructing accordingly his/her programs. Finally, in the open-practice approach, no help is provided to the learner and he has to identify possible strategies by himself for solving the exercises.

Finally, the *assessment part* incorporates tests and exercises trying to identify the outcome of the selected teaching methods and to construct the profile of the end-user. This part can be used by teachers to evaluate the progress of their students, or by students to evaluate themselves and understand their progress. The assessment of each lesson should be completed within 45 min and if the available time runs out without answering more than 50% of the exercises the lesson is considered uncompleted and the student has to do the assessment again. Users can evaluate the assessment methods and procedures and all comments are saved to the knowledge base as well for future reference and extraction of statistics.

A use case-diagram of the functionality of the tutoring module is shown in Fig. 2.

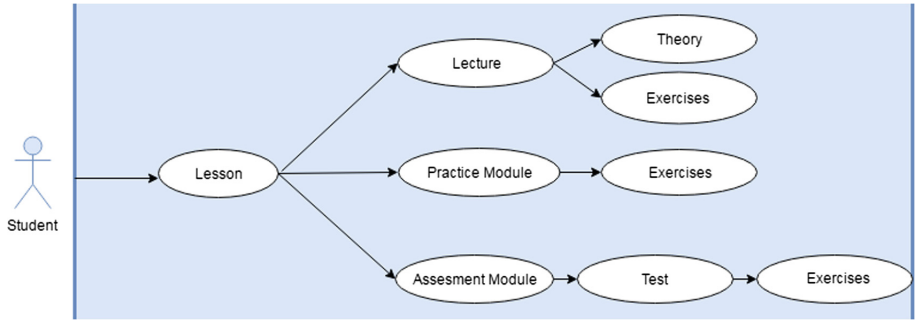

Fig. 2. The Use-case diagram of i-Prolog

3.2 The Student/Learner Module

The student module incorporates the cognitive processes (e.g. information retrieval, calculation and problem solving), the meta-cognitive strategies (e.g. learning from errors) and the psychological attributes (e.g. developmental level, learning style, interests) for each student. As such, the student module represents the learner's emerging knowledge and skills. Information such as learning preferences, past learning experiences may also be relevant for adapting the teaching process. It also records the learner's errors and misconceptions. It continuously monitors and assesses student

performance and updates accordingly the student model. This module provides the dynamicity in student-centered tutoring personalizing the teaching process. In order to do so, information for each unique user is used (his/her email) that are registered by the administrator of the system.

3.3 The Teacher Module

The teacher module enables a distant teacher to communicate with the students using messages, providing him with a complete view of a student's progress through the system and with statistical information. The teacher has the ability to detect the mistakes of his/her students, their selections and their comments and to observe the student's assessments. A use-case diagram of the functionalities of the teacher module is shown in Fig. 3.

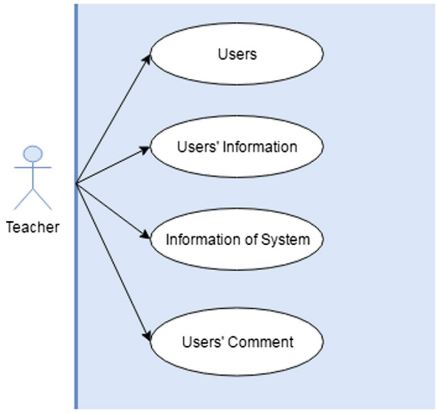

Fig. 3. The Use-case diagram for the teacher module

3.4 The Knowledge Base (KB) and the KB Update APIs

The knowledge Base (KB) contains knowledge related to the subject matter, i.e. Prolog programming. It contains knowledge about Prolog programming and learning in order to make inferences and/or to solve problems. In addition, it contains explanations for problem solutions, providing also alternative explanations for the same topic. Furthermore, it includes a user manual for using the system.

3.5 The User Interface

The user interface provides the means for the students and teachers to interact with the system through a graphical user interface. Some screenshots of the GUI are show in Figs. 4 and 5.

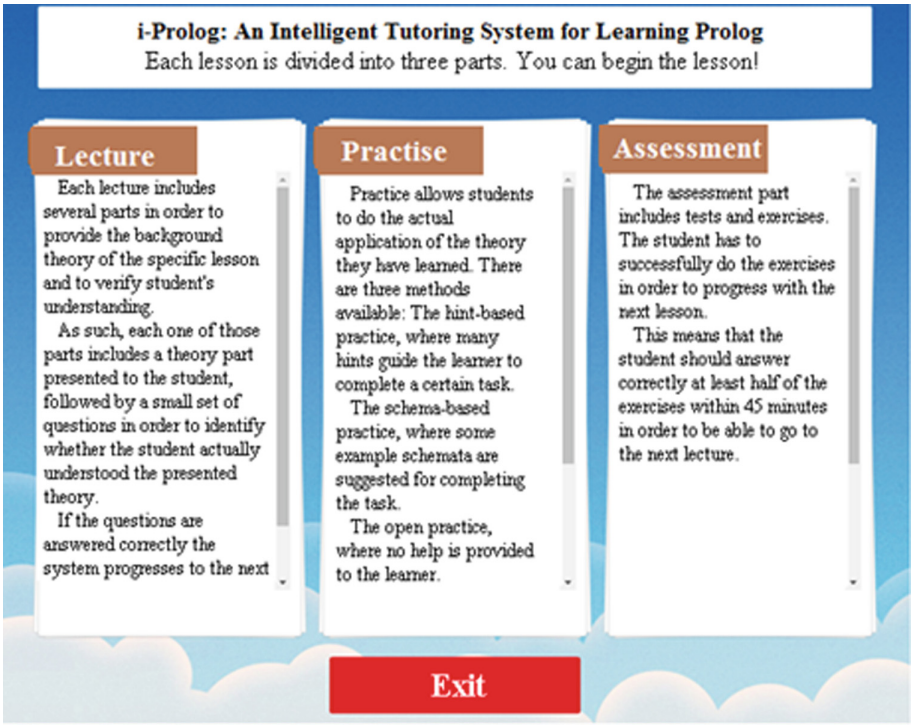

Fig. 4. Selecting lecture, practice or evaluation

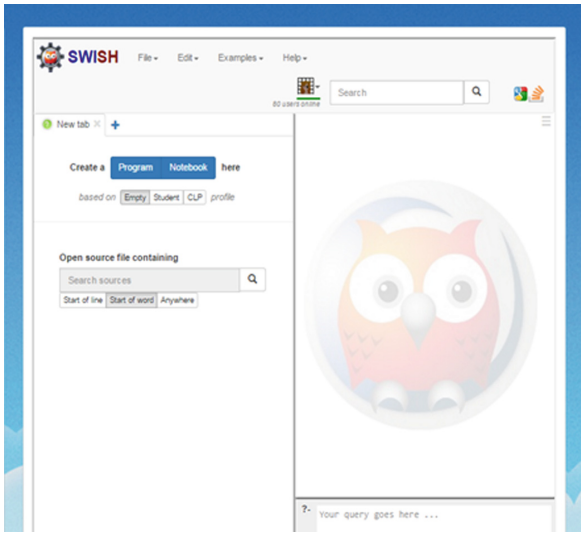

Fig. 5. Practicing using the online Prolog compiler

4 Evaluation

In order to evaluate our system, we conducted a user-study at the Technological Educational Institute (T.E.I) of Crete. More specifically, we made a short introduction of our system to 15 students of the Knowledge Systems undergraduate lesson. Then guided scenarios were executed by the students. At the end, they had to complete an evaluation questionnaire followed by a structured interview.

Questionnaire's results: The questionnaire was designed using Google forms and contained both closed-ended (limiting the answers provided) and open-ended questions. According to the questionnaire's result all users (100%) were able to quickly understand and use our system. In addition, 67% of them reported that it was quite intuitive to identify the theory required for practicing and the corresponding evaluation assessment. 93% of them identified that the examples provided were really good for understanding the theory. 86% of the students answered that the available practice methods were adequate in all occasions. Using an overall five likert scale for the quality of the system the results are shown in Fig. 6, showing the high quality of our solution.

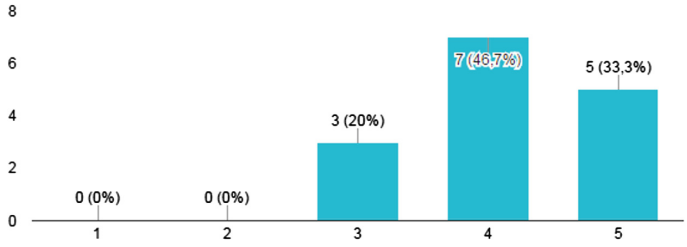

Fig. 6. Overall evaluation of the quality of the system (5 the best)

Structured Interview results: According to the structured interviews conducted, all students liked the system and found it really useful and usable. They considered that the system offers a pleasant environment for learning Prolog and they reported that there are no unexpected errors or strange behavior. Furthermore, the students made some proposals for improving the real-time communication between the teacher and the students in the practice mode and suggested to improve the suggestions provided after the error-detection in the Prolog code.

5 Conclusions and Future Work

i-Prolog is an e-learning system used for student-centered teaching. It focuses on the learner and gives specific attention to provide knowledge by letting the students themselves to discover the learning style that best suits them. As such, the tool helps students to gain a positive attitude towards the learning process. The use of such a system can offer significant benefits for both the teacher and learners due to its ability to be used in many ways and to adapt according to individual student profiles. It can be

applied to both traditional education and distance learning. In traditional education, it can be a tool enriching the teaching process, while it can be used remotely to replace completely the physical presence of the teacher. The main objective of this work is to enhance the interest of students and teachers and to become a precursor for new research that will evolve and improve its construction and its use.

We have to admit, that there is a margin of improvement from both the computer and the education side. Until now, there are systems that offer asynchronous communication with the teacher who manages the system. The new challenge is to turn the ITS into a system that will enable communication between not only a student and an educator but among all students as well. The aim of this improvement is to integrate the theories of cooperative and socio-cultural learning that are not yet included in these systems. This of course, will add new ways of practicing, where students could be divided into groups and performing collaboratively the training through constructive communication. As such, the teaching will be transformed to an interactive game, available online, designed to make the learning process more enjoyable and effective. Finally, it is in our immediate plans to exploit the currently implemented tool in order to further educate patients for the management of their disease exploiting our intelligent recommendation [6–8] and see how this tool can be adapted in a versatile new domain.

References

1. Bielikova, M.N.: A schema-based approach to teaching programming in Lisp and Prolog. PEG (2003)
2. Corbett, A., Anderson, J.: Locus of feedback control in computer-based tutoring: impact on learning rate, achievement and attitudes. In: SIGCHI 2001, pp. 245–252 (2001)
3. Gerdes, A., Heeren, B., Jeuring, J., van Binsbergen, L.T.: Ask-Elle: an adaptable programming tutor for haskell giving automated feedback. IJAIED **27**(1), 1–36 (2016)
4. Giannoulis, M., Marakakis, E., Kondylakis, H.: Developing a collaborative knowledge system for Cancer Diseases, IEEE CBMS (2017)
5. Graesser, A.C., Conley, M.W., Olney, A.: Intelligent tutoring systems. In: APA Handbook of Educational Psychology. Department of Psychology & Institute for Intelligent Systems, pp. 1–54 (2010)
6. Kondylakis, H., Kazantzaki, E., Koumakis, L., et al.: Development of interactive empowerment services in support of personalized medicine. eCancer Med. Sci. J. **8**, 400 (2014). https://doi.org/10.3332/ecancer.2014.398
7. Kondylakis, H., Koumakis, L., Kazantzaki, E., et al.: Patient empowerment through personal medical recommendations. Health Biomed. Inf. **216**, 1117 (2015)
8. Kondylakis, H., Koumakis, L., Ruping, S., et al.: PMIR: a personal medical information recommender. In: Proceedings of Medical Informatics Europe (MIE), vol. 205, p. 1193 (2014)
9. Looi, C.K.: Automatic debugging of prolog programs in a prolog intelligent tutoring system. Instruct. Sci. **20**, 215–263 (1991)
10. Marakakis, E., Kondylakis, H., Papadakis, N.: A knowledge-based interactive verifier for logic programs. Innov. Knowl. Intell. Eng. Syst. **18**(3), 143–156 (2014)
11. Moon-Chuen, L.: An augmented prolog programming for tutoring applications Environment Components. In: AIE, pp. 898–906 (1990)

12. Rivers, K., Koedinger, K.R.: Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. IJAIED **16**(1), 37–64 (2015)
13. Robins, A., Rountree, T., Rountree, N.: Learning and teaching programming: a review and discussion. Comput. Sci. Educ. **13**, 137–172 (2003)
14. Skinner, B.F.: About Behaviorism. Vintage, New York (2011)
15. Vlahavas, I.P., Sakellariou, I., Futo, I., Pasztor, Z., Szeredi, J.: CSPCONS: A communicating sequential prolog with constraints. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) SETN 2002. LNCS (LNAI), vol. 2308, pp. 72–84. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46014-4_8
16. Webb, G.I.: Inside the unification tutor: the architecture of an intelligent educational system. In: ASCILITE, pp. 677–684 (1991)
17. Webb, G.I.: The unification tutor - an intelligent educational system in the classroom. In: ASCILITE, pp. 408–420 (1989)
18. Wikipedia Article: intelligent tutoring system. https://en.wikipedia.org/wiki/Intelligent_tutoring_system. Accessed Aug 2017
19. Yang, S.M.: Approaches for learning prolog programming. Innov. Teach. Learn. Inf. Comput. Sci. **6**, 88–107 (2007)