Concept of an Interactive Web Portal for Teaching Prolog

Grzegorz J. Nalepa and **Igor Wojnicki**
Institute of Automatics,
AGH – University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
gjn@agh.edu.pl wojnicki@agh.edu.pl

Abstract

The paper presents observations concerning teaching Prolog to computer science students. Basing on these experiences an interactive web-portal is proposed to support teaching. Some guidelines for portal features are given. A prototype implementation based on DokuWiki is also described.

Introduction

This paper discusses practical problems encountered in teaching Artificial Intelligence and Prolog to computer science students. One of the main issues with teaching Prolog is a large conceptual difference between declarative and object-oriented programming paradigms, that the older students are used to. In general, students find learning Prolog a hard experience. Compared to imperative programming, declarative programming requires switching to a different way of thinking Because teaching AI is usually offered as a high-level course (for graduate students), at that time students are already well accustomed to the imperative programming paradigm, and switching to the declarative one usually poses a real challenge.

This paper proposes a web-based portal providing a learning environment for Prolog, that tries to address the problems students have. In the portal a practical Prolog course is contained, along with several hundred examples of Prolog programs. The portal allows students not only to obtain information, but also provides means for expressing feedback in the learning process. The students can write comments and suggest solutions to the problems they encountered. The portal is based on the wiki paradigm and it is implemented using a PHP-based DokuWiki solution. This wiki combines a number of presentation options with advanced versioning features. It is aimed at supporting the courses in the Institute of Automatics, AGH University of Science and Technology.

Teaching AI with Prolog

Teaching Artificial Intelligence (AI) (Russell & Norvig 2003; Negnevitsky 2002) is a non-trivial task. It is a domain combining knowledge from multiple fields, including computer science, psychology, mathematics, etc. Some of the achievements from the field of AI have been adopted in applied computer science (CS). However, AI still remains a domain of an active development of non-classic approaches to problem solving. So, building AI systems may require a number of different skills.

Because of this complexity, teaching AI poses a number of challenges. Surprisingly, teaching applied AI to CS students may sometimes be more difficult than to other engineering students. CS students come to the AI class with a number of well founded ideas about "programming", "design", or "knowledge". They are also very well equipped with efficient programming tools to solve "classic CS" problems. However, problems considered in AI are often more general or more abstract. CS students are mostly convinced that solving problems is "just programming", and programming is about writing and combining functions, objects that are in the end executed sequentially. However, AI efficiently combines a number of non-classic "programming" approaches, and does not enforce a single paradigm, quite often combining imperative and declarative approaches.

These problems are especially visible when teaching non-classic programming languages such as Lisp or Prolog (Covington, Nute, & Vellino 1996; Bratko 2000). Compared to C/C++/Java, Prolog is a very high-level language, based on completely different concepts. One of the main problems is the the extensive use of recursion in Lisp and Prolog. Students are not very familiar with it, at least not as an universal programming technique. They think in terms of iterative programming and loops. The difference between Prolog and these languages makes it almost impossible for students to learn it as "yet another programming language".[1] This approach does not work, since in Prolog, there are no keywords, functions, methods, or objects, it even does not use a clear input/output concept. It turns out to be extremely expressive, though. Prolog operates on a more abstract *knowledge level* (Newell 1982), while CS students mostly think on a *data level*. Similar conceptual problems are also related to some other non-classic programming paradigms, e.g. in functional programming. In general, students have trouble with switching to other programming paradigms.

[1] This approach could work quite well for C/C++/Java/PHP, even though in fact students should rather be taught OOP not as an "extension" of procedural programming.

It was observed that some students have less problems with grasping this different approach. Moreover, they tend to discuss the approach with others. This gives a hint that a highly collaborative information sharing platform could help out these weaker students to keep pace with the better ones.

Lessons Learned

Based on the authors experiences, to motivate students and to make them interested in the subject of learning Prolog and its applications the following guidelines can be applied. They are based on blended learning approach, combining e-learning, lectures and lab activities. These are:

- a highly interactive learning experience,
- ability to extend the learning experience with e-learning tools,
- capture student in-class observations and incorporate them into the course syllabus,
- share the above observations with all students,
- provide illustrative examples,
- show interesting and advanced applications, solutions to real-world problems,
- and last but not least, make it fun.

Capturing in-class student observations and sharing them with other students improves the teaching process. Declarative programing is mostly about thinking not implementing. While implementing is something the students are accustomed to, thanks to the imperative programing courses, thinking about the problem to solve, poses a real challenge to some of them.

Usually there are a few bright students who enjoy the declarative approach and they are fully capable of such programming. Sharing their experience with others is a precious asset. Making it persistent in the form of a knowledge base for the course is an asset other students can use.

The course knowledge base should consist of: theoretical information about Prolog, illustrative examples with varying complexity, supported by comments, thoughts and improvements made by other students, real-world applications, including solving problems Prolog is designed for. The knowledge base should be accessible through an interactive, multi-user, multi-access environment. Creating such an environment makes the learning process more stimulating, and since students get support from their classmates, it improves the overall experience.

Gathering such knowledge about programming has several requirements. A learning environment should provide:

- version control – it should be always possible to restore information which has been overwritten or changed in a wrong way,
- search capabilities – searching for topics, problems, solutions, and explanations; encyclopedia-like features,
- syntax highlighting – to visualize programming language code in a more user friendly form,

Last but not least it is important to provide a system built in the native (polish) language. Most of the tutorials and examples for Prolog are in English. Even though students know and use English, the learning experience can be largely improved, by providing examples using polish, since Prolog does not enforce any English keywords anyway.

To address the problems identified, and to meet the requirements, in the next section a solution is proposed.

Proposed Solution: Wiki-based Teaching

Considering challenges mentioned before the concept of an *interactive* portal for teaching Prolog has been formulated. The basic idea is to support the Prolog class with an integrated web-based tool, meeting the following requirements:

- it is able to provide effective feedback in the teaching process,
- it captures the knowledge about problem areas discovered by the students, during the class,
- it provides a platform for distributed authoring, for several teachers that teach the same class to different students,
- it allows students to record their ideas, and possibly extend the class.

All these requirements lead to choosing a collaboration environment based on the wiki concept.

Wiki Systems

The Wiki concept emerged in the 90's. The main idea was to create a simple and expressive tool for communication and knowledge sharing. Thus, the main features follow this idea.

A wiki system is mainly a collaboration tool. It allows multiple users to access, read, edit, upload and download documents. It has a regular client-server architecture. Documents are text-based, enriched with so-called wiki markup. It is a simplistic, tag-based, text only language which allows the user to annotate text with information regarding its structure and presentation. The tags allow users to make sections, subsections, tables, items and other typographic and structural operations. A wikitext remains human readable, tags are intuitive and easy to learn.

Each document is uniquely identified by a keyword, which makes the wiki concept similar to the encyclopedia concept. Furthermore a document, in addition to typographic tags mentioned earlier, can contain hyperlinks to other documents. A hyperlink is a document name enclosed by a link tag. The wiki allows users to upload images, as well as other files and link them together.

Wikis are mostly web-based. The web interface allows users to access, render and edit wiki pages. Depending on the particular solution there might be access control and authorization mechanisms implemented as well. A wiki system is usually based on some server side processing technologies providing the web application, and optionally a database backend. As a frontend a web browser is used.

One of the most important features of a wiki is an integrated version control. Page modifications are recorded. At any time a user can access any previous version of any page.

It is worth pointing out that Wiki Systems currently blend with Content Management Systems (CMS). Some CMS provide Wiki functionality while some Wikis evolve into CMS. Similarly Wikis are more and more often merged with *e-Learning* systems to support collaborative knowledge gathering and sharing.

The Dokuwiki System

One of the most interesting Wiki systems for developers is DokuWiki (`wiki.splitbrain.org/wiki:dokuwiki`). It is designed to be both easy to use and easy to set up. DokuWiki is based on PHP and does not require any database backend. Pages are stored as versioned text files which enables easy backup-restore operations. It allows for image embedding and file upload and download. Pages can be arranged into so-called namespaces which act as a tree-like hierarchy similar to directory structure. It also provides syntax highlighting for in-page embedded code for programming languages such as: C/C++, Java, Lisp, ADA, PHP, SQL and others, using GeSHi (`qbnz.com/highlighter`).

Furthermore it supports extensive user authentication and authorization mechanisms including Access Control Lists (ACL). Its modularized architecture allows the user to extend DokuWiki with plugins which provide additional syntax and functionality. The most popular plugins provide: user and ACL management, blog, gallery of pictures, discussion board, calendar, and LaTeX symbols rendering.

System Prototype

A prototype of the *Prolog Wiki System* is currently (fall 2007) being implemented in the Institute of Automatics, AGH, reachable at the address `https://ai.ia.agh.edu.pl/wiki/prolog`. The configuration includes: DukuWiki system, running on the Apache webserver, version 2, with the PHP5 environment, in the Debian/GNU Linux 4 "Etch" operating system. The system includes the Prolog syntax highlighting module, developed by students.

The *Prolog Wiki System* stores domain information about AI and Prolog. It includes problem statements, their solutions, examples, and lab assignments as well. Both teachers and students are allowed to modify certain parts of the wiki. All changes in the wiki are subject to version control. They can also upload and download files. Prolog language code embedded into wiki pages is automatically syntax highlighted, by a dedicated GeSHi module for Prolog, developed by students (see `https://ai.ia.agh.edu.pl/wiki/prolog:geshi_prolog`).

Basic E-Learning

The prototype provides basic e-learning features. Students can study after or before the class some of the assignments are given remotely using the system

Capturing the Experience

The wiki system is interactive. It is open to the students during, and after class. So, it is possible to capture student in-class observations and incorporate them into the course syllabus on-line. Students can also write comments to the examples, or provide alternate solutions to the examples.

Programing Examples

In the 2006 and 2007 class of "Knowledge Engineering Methods" several teams of students were supposed to select, describe, and test in SWI-Prolog some of the illustrative examples of Prolog code, freely available on the Internet for students and teachers. These projects resulted in selecting about 400 examples of Prolog code. Currently the database of examples is being integrated with the system.

The code library has been build using examples from several Prolog books (in cases where the examples where freely available), including (Bratko 2000; Covington, Nute, & Vellino 1996; Sterling & Shapiro 1994), as well as Prolog tutorials. So far, the examples have been divided into three general groups: learning the Prolog language, solving classic AI problems, knowledge representation and expert systems. The idea is to work with students on the selection and categorization of the examples. The students annotate the example base, providing information which examples proved useful during the class. This experience helps in workings with subsequent student groups.

Advanced applications

Besides some "classic" Prolog topics, several advanced issues are being presented in the system. Currently two main themes are: integration, and business rules. Exercises on Prolog integration with other languages, including ANSI C, C++, and Java, are based on standard packages provided by the SWI-Prolog. The business rules topic is related to the practical design and implementation of rule-based logic for business applications. These exercises give students some ideas about possibilities of combining the features of Prolog with existing solutions.

Make it fun: Robotics

To give students a real-world application of the Prolog language an intelligent control theme emerged. The main idea is to use AI techniques to program and control mobile robots. There is a multi-layer architecture used: low level control primitives in an imperative programming language, Intelligent Control based on Prolog. Currently two platforms are being used: Hexor and LEGO Mindstorms.

HexorII is a mobile robot platform provided by Stenzel (`stenzel.com.pl`). The robot moves like a scorpion, using three servos: one for tilt, two for forward and backward leg moving. With this kind of construction HexorII executes the fast insect movement algorithm.

LEGO Mindstorms NXT is a standard platform for teaching robotics. It is being introduced in the class in Fall 2007. Currently, the possibilities of controlling Mindstorms with Prolog are limited. Several student projects have been started to provide Prolog support for Mindstorms, these are:

- providing a low-level communication module using USB or Bluetooth; three main options are considered: direct Prolog serial port programming, using an ANSI C based stack integrated with Bricx (`bricxcc.`

sourceforge.net), and integration with the LeJOS (lejos.sourceforge.net) iConnect module,

- providing a middle-level Prolog API for controlling NXT on the basic component level, that is motor and sensor communication, a compatibility layer with LeJOS is considered here,
- a high-level robotic API to control navigation, etc.

The wiki system proved to be quite helpful in coordinating multiple student groups working on the Mindstorms API.

The Big Picture

Interactions between users, teachers and students, and the prototype system is given in Fig. 1. Dashed arrows indicate that particular element is derived from or it is related to another one. All interactions take place in the Wiki system. There are AI Classes and AI Project Subjects prepared by teachers. Students are welcome to modify contents of the classes and add comments and proposals. These comments/proposals always regard particular topic: Prolog Classes, Advanced Applications, Intelligent Control Classes, or Prolog Programming Examples. Students are also welcome to document AI Projects they take. These projects are directly derived from the AI Project Subjects specified by teachers.

Currently the Prolog Wiki System is swallowed by *aiWiki*, a more general system hosting, in addition to Prolog, general AI topics and serving as a collaboration platform for several research projects.

Related Research

In this paper a personal perspective on teaching Prolog to CS graduates in the Institute of Automatics at AGH University of Science and Technology is given. Of course the topic of difficulties with teaching Prolog is not new.

Some of the guidlines and other teachers' experiences have been published and discussed numerous times (e.g. (Brna, Pain, & du Boulay 1990; Hietala 1993)). However, it is worth noting that teaching techniques are dependent on the supporting tools that are available. There are some interesting studies on how choosing different tools can improve the learning process (Yang & Joy 2007). The particular choices depend on both student and teacher preference.

Currently there is a large number of specialized e-learning tools, with *Moodle* being a prime example. However, it was decided that a simple wiki-based system is a better solution for this class. Extra features provided by Moodle were not useful enough to justify the large overhead and complexity imposed by this system. The "less-is-more" approach of the DokuWiki along with its extendibility via plugins proved to be much more efficient.

Evaluation

The main AI wiki system was started in early 2007 to support a research project. The prototype Prolog portal was implemented in fall 2007. So far, the system supported the:

- the AI languages class (lab) of 40 students, Fall 2007,
- the Knowledge Engineering projects, 40 students, Spring 2007,
- the Knowledge Engineering languages and tools class (lab), 60 students, Spring 2007.

It can be observed, that while students enjoy the idea of an interactive portal, many of them are not very active in enriching it if they are not obliged to. On the other hand, during the practical coordination of KE projects, were individual students where supposed to record there progress in the wiki, their responsiveness was way better then with the regular approach (meeting in class and email contact).

When it comes to the practical enrichment of the Prolog examples base and the lab the results are inconclusive so far. It seems that only the best and most motivated students spend their time working in the wiki. While their insight is valuable to the teachers and other students, some ways of encouraging the others are tested.

Future Work

There are several subprojects in progress related to teaching AI. They are focused on practical applications of AI techniques and Prolog in particular. They involve both faculty and graduate students. Collaboration between students working on the projects and faculty members is provided through aiWiki. These projects serve also as examples for students learning Prolog, proving its practical relevance.

Prolog Server Pages

The project aims at deploying and using a technology called Prolog Server Pages (PSP). It regards a client-server architecture based on HTTP. It is a server-side scripting language based on Prolog. PSP is embedded in HTML documents and interpreted as a Prolog program. The output is then sent to the client (i.e. a browser) together with the native HTML code. There are several approaches to PSP available. Some of them are: Prolog Server Pages by Mauro Di Nuzzo (www.prologonlinereference.org/psp.psp) and SWI Prolog HTTP Server (www.swi-prolog.org).

Using Prolog as such a scripting languages turns to be interesting for many graduate students. It also turns to be efficient in terms of programming and XML parsing.

Extending Wiki with Prolog

To support the AI teaching process an idea of a Semantic Wiki based on Prolog has emerged. It would support running Prolog code on the web server while rendering a wiki page. Contents of such a page consist of a human-readable text, and optionally images, attachments etc, and machine-readable and automatically interpreted knowledge expressing what the page is about.

To support Prolog programming within the wiki there is an ongoing prototype implementation of a Prolog inference engine embedded into a DokuWiki system working as its extension. It is called *Prolog DokuWiki*.

As a result, in addition to text-based human-readable contents, there are Prolog clauses embedded into wiki pages.

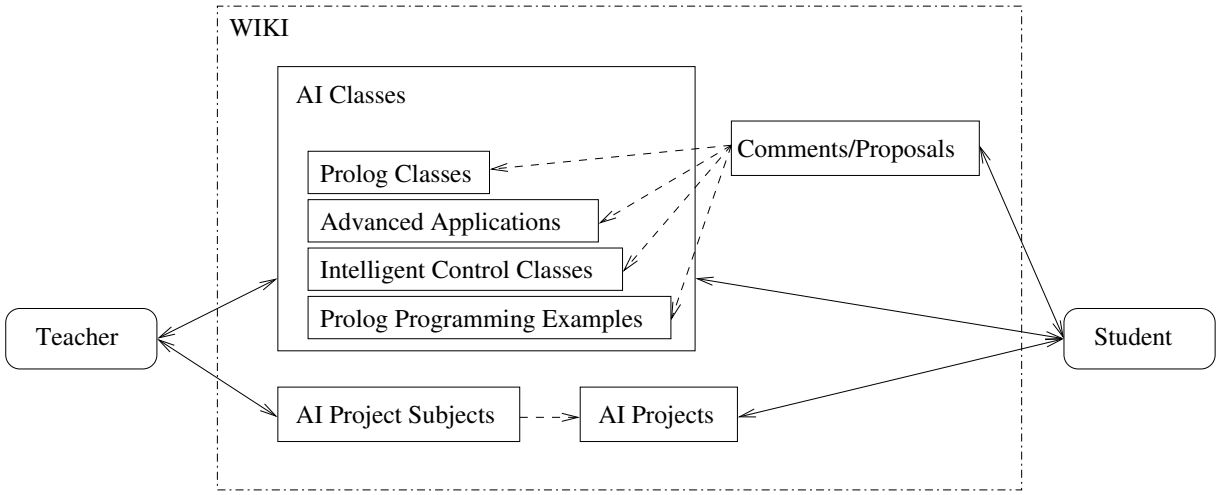

Figure 1: Interactions between users and the prototype system

These clauses can be automatically interpreted upon requesting the page. Results of the interpretation (inference) process are directly rendered into the page.

This constitutes a Semantic Wiki System based on Prolog. It is similar, to some extent, to the semantic wiki systems currently available (Semantic MediaWiki (`semantic-mediawiki.org`), IkeWiki (`ikewiki.salzburgresearch.at`), SweetWiki (`argentera.inria.fr/wiki`)). Other semantic wiki systems use XML to annotate gathered information semantically while Prolog DokuWiki uses Prolog clauses. Furthermore, the clauses are interpreted upon displaying a given page.

The extension introduces a new element indicated by a tag called `prolog`. Any text within the element is treated as Prolog clauses, it is interpreted by an externally launched Prolog inference engine. As the inference engine SWI-Prolog is used. Upon rendering a page with a `prolog` element, the wiki system launches the inference engine which processes clauses within the tag. Standard output of the inference process is displayed in place of the element.

There are special predicates which allow populating the knowledge base with clauses from arbitrary chosen pages or namespaces. There is a `wiki/1` predicate defined which triggers the inference engine to interpret clauses embedded within other wiki pages. If the first argument is a valid wiki page, the predicate browses it and interprets (consults) all clauses within `prolog` tags on this page. If the argument is a namespace it browses all pages from this namespace and interprets all clauses found within the pages. There is another predicate `wiki_recursive/1` which interprets clauses from all pages in the given namespace and all namespaces within it recursively.

Summary

This paper presents some observations on teaching Prolog to Computer Science students. Based on authors' experiences, a concept of an interactive Wiki is proposed. The Wiki aims at capturing student experiences and comments and provides a basic e-learning solution.

Currently a prototype of the system is running at the Institute of Automatics at AGH University of Science and Technology. The system is being used for the Fall 2007 course in AI languages, as well as the Spring 2008 Knowledge Engineering class. So far, it helped a lot in improving both the learning and teaching experience.

Acknowledgements The paper is supported by the *HeKatE* Project funded from 2007–2009 resources for science as a research project.

References

Bratko, I. 2000. *Prolog Programming for Artificial Intelligence*. Addison Wesley, 3rd edition.

Brna, P.; Pain, H.; and du Boulay, B. 1990. Teaching, learning and using prolog: Understanding prolog. *Instructional Science* 19(4–5):247–256.

Covington, M. A.; Nute, D.; and Vellino, A. 1996. *Prolog programming in depth*. Prentice-Hall.

Hietala, P. 1993. Teaching ai through prolog programming techniques. *Comput. Educ.* 20(1):133–139.

Negnevitsky, M. 2002. *Artificial Intelligence. A Guide to Intelligent Systems*. Harlow, England; London; New York: Addison-Wesley. ISBN 0-201-71159-1.

Newell, A. 1982. The knowledge level. *Artificial Intelligence* 18(1):87–127.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition.

Sterling, L., and Shapiro, E. 1994. *The Art of Prolog*. Advanced Programming Techniques (Logic Programming). MIT Press.

Yang, S., and Joy, M. 2007. Approaches for learning Prolog programming. *Innovation in Teaching And Learning in Information and Computer Sciences* 6(4).