A Web-Based Education System for Predicate Logic

Ioannis Hatzilygeroudis, Christos Giannoulis, Constantinos Koutsojannis
University of Patras, Dept of Computer Engin. & Informatics
26500 Patras, Hellas (Greece)
&
Research Academic Computer Technology Institute
P.O. Box 1122, Patras, Hellas (Greece)
{ihatz/giannoul/ckoutsog}@ceid.upatras.gr

Abstract

In this paper, we present a web-based system teaching predicate logic as a knowledge representation and reasoning language. The system is adaptable in the sense that it allows the students to choose their own way of using it. Students can evaluate themselves, by selecting the complexity and the difficulty level of the exercises. Another interesting point of the system is its open exercising facility, by which the students can try any conversion of a FOPC formula to Clause Form. This is achieved by calling LISP code, which is part of an automated theorem prover. Incorporation of LISP code into the hypermedia application was an interesting implementation problem. An initial evaluation of the system showed encouraging results as far as its usability and learning are concerned.

1. Introduction

Knowledge Representation & Reasoning (KR&R) is a fundamental topic of Artificial Intelligence (AI). A basic KR language is First-Order Predicate Calculus (FOPC), the main representative of logic-based representation languages, which is part of almost any introductory AI course. To make automated inferences, Clause Form (CF), a special form of FOPC, is used in conjunction with Resolution Principle (RP), a very powerful rule of inference. Students find difficulties in various aspects of using FOPC as a knowledge representation and reasoning language. Two of them are (a) converting natural language (NL) sentences into FOPC formulas and (b) converting complex FOPC formulas into CF.

There are several systems, like Plato [1], Logic Toolbox [2] etc (see e.g. in http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html#list for an account) that are characterized as logic educational software. However, most of them have been developed at Philosophy Departments and deal with how to construct formal proofs mainly using natural deduction rules, restricting themselves to propositional logic (PL). PL is weaker than FOPC: there is no ability to use variables. An interesting and advanced case is Logic-ITA [3], which is an intelligent teaching assistant system for Logic. It also deals with prepositional logic in the same sense as above, but it is addressed to both students and teachers and uses intelligent techniques to automatically adapt to their needs. So, those systems are not concerned with how to use predicate logic as a KR&R language.

To help the students and the tutor in our Department, we constructed a web-based system to assist learning and teaching logic as a KR & R language (course 451: AI). The system focuses on the above two difficulties of students. A student can specify the level of difficulty and the level of complexity of the exercises to test him/herself at his/her own pace. Also, the system offers some openness as far as FOPC to CF conversion is concerned. The user can try any FOPC formula and test each conversion step by him/herself. So, the system can be characterized as adaptable to the student needs.

2. System architecture and functionalities

The architecture of the system is depicted in Figure 1. It comprises three main units: the *Hypermedia Application* (HA), the *LISP Application* (LA) and the *Database* (DB). Students are the users of the system. A student has to identify him/herself before logging in the system and can access HA through a Web-browser. Then the Web Server (in fact IIS) is responsible for the interaction between the user and the system.

HA is actually the tutoring part of the system. A student is able to study theory about various concepts and procedures related to FOPC as a KR&R language, look at a number of solved examples and try some exercises, to test his/her knowledge. All about examples and exercises are stored in the DB. So, there is an interaction between the HA and the DB during a learning session. In some cases, students can try their own examples or exercises. In those cases, LA is called to compute the results, which are then displayed back to the student, through the HA.

Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04)
0-7695-2181-9/04 $20.00 © 2004 **IEEE**

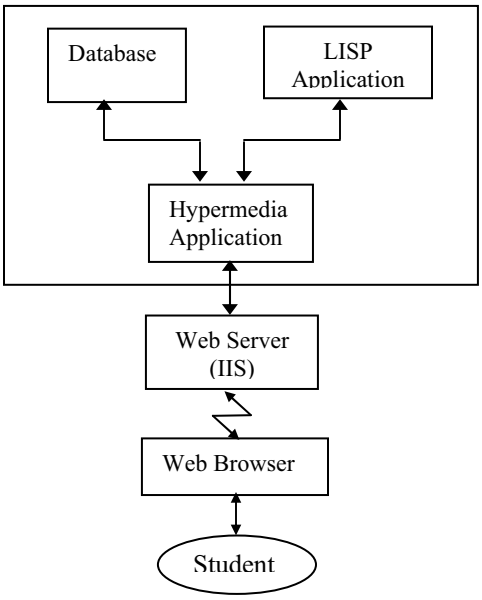

Fig. 1. System Architecture

3. The Hypermedia Application

3.1 Content and Its Presentation

The Hypermedia Application (HA) presents the learning content to the students. The interface of the HA consists of two areas: *navigation area* and *content area*, and two bars: *info bar* and *tool bar* (see Fig. 2). The info bar, at the top of the screen, displays the local time and date, the time the user has been connected to the system so far and the number of currently connected users. The tool bar, at the bottom of the screen, contains a number of links: logout, password change, initial page, help page, technical support page and communication (with the tutor). The content area is the main area, where the learning content is presented, and resides at the center and the right part of the screen.

The navigation area, at the left side of the screen, displays the contents the students can deal with, in the form of a pop-down tree/hierarchy (see Fig. 2). Part of the tree is presented in Fig. 3.

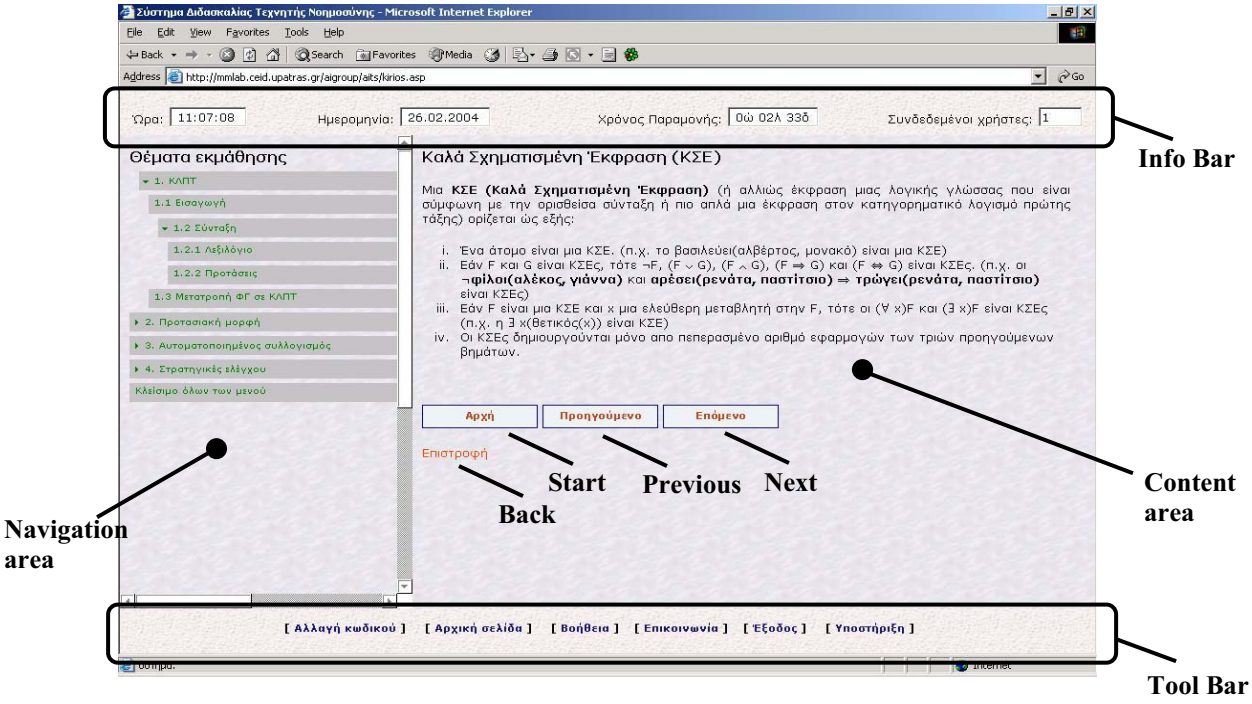

Fig. 2. Hypermedia Application User Interface

Topics lower down in the hierarchy are less complex than those higher up in it. The leaves of the tree (in rectangles) correspond to *simple topics*. Each simple topic corresponds to a *topic page*, which is an ASP page. That is, only content about simple topics is displayable. The topic page of the selected simple topic is currently presented in the content area. Each topic page deals with a number of *concepts*. More specifically, it contains an *ordered list* of concepts. Each concept is linked to the corresponding *concept page*.

The learning method (implicitly followed) is based on the traditional theory-examples-exercises paradigm (although the user can follow his/her own method). That is, for each topic, the theory is first presented. Then, some

examples are given. Finally, the student is called to solve some problems or make some exercises. Theory consists in presenting a number of concepts. Those concepts are presented in a simple-to-complex way. That is, the simple concepts are presented first and the complex concepts (that require the knowledge of one or more simpler concepts) are presented afterwards. This is depicted in the ordered list.

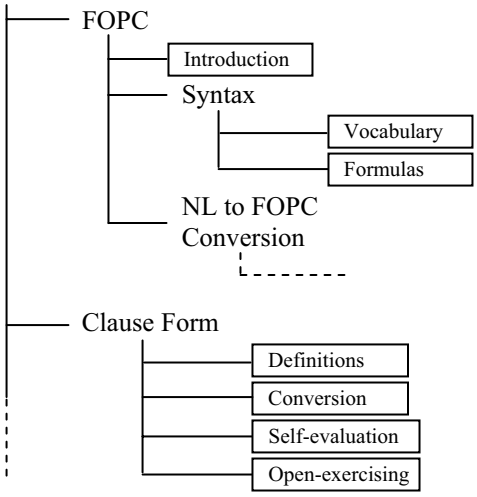

Fig. 3. Part of the Content Navigation Tree

Furthermore, the student can review a previous concept at any time. The student is also not forced to follow our way of teaching, but can make his/her own choices for studying. For example, a student can jump to complex concepts without taking a look at simpler ones. In many concept pages there are links to other concepts that are *prerequisite* to the concept of the page. So, the student, if needed, can recall the theory about the prerequisite concepts. After having looked at the recalled theory, the student can return back to where was before and go on with his/her studying.

3.2 Self-Evaluation Facility

A student is able to test what has learnt so far, by taking some exercises. The student has the opportunity to set his/her requirements for each exercise. Those requirements are passed as a query to the DB and the system responses with an appropriate exercise (retrieved from the DB). The student can try to solve the exercise by him/herself and then check the answer. This is the way a student can do a self-evaluation on NL to FOPC conversion and on FOPC to CF conversion.

In the first case, the student selects a difficulty level (1 to 5) and specifies the complexity level of the exercise. The complexity depends on how many of the vocabulary types: constant, variable, function and quantifiers, are present in the resulted FOPC formula. The student has to select at least one of them. Then the system picks up a clause in NL and asks the student to convert it into FOPC. The student works with it and when is ready checks whether his/her answer is similar to any of four (4) possible answers given by the system. If the student checks a wrong one, an explanation is provided why it is wrong. Then the correct answer appears.

In the second case (FOPC to CF), the student selects a difficulty level (from 1 to 5) and the system responds by returning an appropriate FOPC clause, to be transformed into CF. Transformation includes six steps: implier elimination, negation reduction, variable renaming and transformation in PNF, Skolemisation and universal quantifiers removal, transformation into CNF, clause extraction and variable renaming.

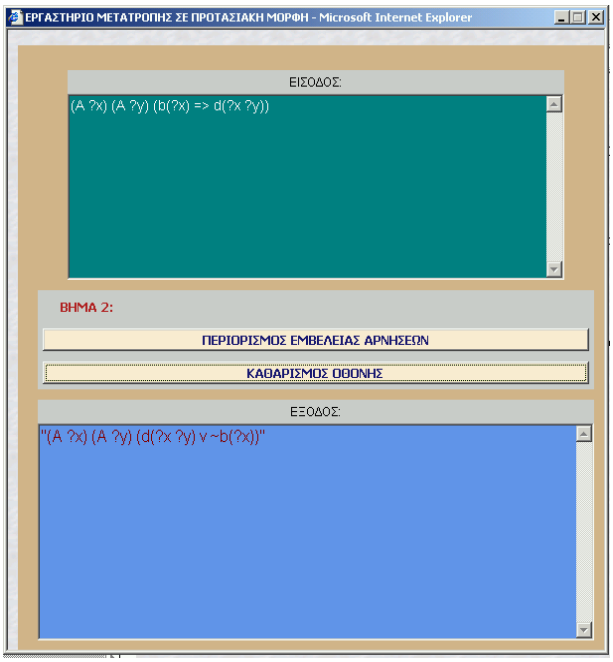

Fig. 4. Open exercising facility interface

Therefore, the answer is constructed in a stepwise manner. In every step the system gives the student the right answer. So, the student is able to check at every step if has made some mistake. If he/she has, is able to go on with the next step, using the correct answer given by the system. In each step, before or after its execution, the student can follow a link to the theory related to that step. So, the student can recall that theory as a help either to construct his/her answer to the step or to verify the answer given by the system. After completing the whole function, that is all steps, the final result, the correct clause in CF, appears.

3.3 Open Exercising Facility

While self-evaluation facility allows a student to evaluate him/herself in a semi-structured way, open exercising facility allows him to do it in a free way. A student can try to convert any FOPC formula to its CF, in a stepwise manner. So, on the one hand, a student can check his/her ability in converting any formula and, on the other, he/she can practise by trying as many formulas as he/she wishes.

This automated conversion is achieved by calling a java applet, which runs LA code (see next section). The interface of the facility includes three areas (see Fig. 4). The upper area is used for FOPC formula input. The lower area is used to display each step's result. Finally, the mid area specifies the step to be executed next.

4. Implementation Issues

According to [4], architectures of web-based tutoring systems can be distinguished in three categories, based on the location where the tutoring functions are performed: the *centralized*, the *replicated* and the *distributed* architecture. In the centralized architecture, all tutoring functions are performed on the server machine, whereas in the replicated one on the client's machine. In the distributed architecture, tutoring functions are distributed between the client and the server. The architecture of our system belongs to the distributed one. Our application server consists of HA and DB and executes its functions on the server. LA is implemented as a Java applet, which is downloaded and executed on the client's machine.

All web pages of the HA have been implemented in ASP. By doing this, we make sure that only authenticated users would have access to the system. Also, ASP provides some features that allow communication with a database. Hence, ASP is a good tool, since the HA needs to interact with the DB. The DB has been implemented in Microsoft Access.

A very interesting part of the implementation was that of LA and its connection to the HA. LA is actually part of an Automated Theorem Proving (ATP) system, called ACT-P, implemented in Common LISP [5]. The problem was how to call a LISP program (functions) from within a Web-based system. To this end, we found a LISP interpreter implemented (by Josef Jelinek) in Java, called G-LISP. G-LISP environment is called LISP Interpreter Java Applet (LIJA) [6]. It was not what we exactly needed, but it was a very good tool, since it could allow us to use LISP on the Web and its source code was available.

The first thing we realized was that not all Common LISP functions were implemented in G-LISP. So, first, we enhanced it with a number of functions. Another problem we faced was to be able to automatically load LA functions, those related to the FOPC to CF conversion. LIJA has a feature that allows to load LISP functions from a file, but requires that the LISP functions are on the client side. This problem has been solved by constructing a C program, which reads LISP functions from a file and returns a file with a Java method. This method is then used in the Java Applet. By doing this, we can load as many functions as needed. The C program runs on the server side and, after the methods are placed in the applet, the applet is compiled.

5. Initial Evaluation

The first version of the system was released in December 2003 and used by the class of the Artificial Intelligence course, in our Department, which consisted of thirty senior computer engineering students. The students had been taught about FOPC as a KR&R language during the course lectures. They were instructed to use the system as follows: login at least three times and make at least (a) three exercises from NL to FOPC conversion, (b) two exercises from FOPC to CF conversion using the self-evaluation facility and (c) two exercises from FOPC to CF conversion using the open exercising facility. Then, they were asked to fill in a questionnaire (similar to that used for the system in [4]), including questions for evaluating usability and learning. The questionnaire included ten questions. Question 1 was of multiple choice and concerned the time needed for a student to adapt to the system. Question 3 was a yes-no (actually agree-disagree) type question and concerned a comparison of using the system and attending a tutorial session. Questions 2 and 4-6 were based on Likert scale (1: not at all, 5: very much) (see Table 1), but they included a 'please explain' request too. Finally, questions 7-10 were of open type and concerned strong and weak points or problems faced in using the system.

Twenty eight students filled in the questionnaire. Their answers showed that the students in general enjoyed learning with the system (Q4, Table 1). Most of them (82%) reported that they needed less than five minutes to start using the system (Q1). Also, they found that the user interface is easy to use (Q6, Table 1).

On the other hand, the students agreed that the system helped them in learning about Logic. The average score in the relevant question (Q2, Table1) was 3.9. Also, due to this fact (as extracted from the 'please explain' section of the question), they suggest the system to the next year students with an average score 4.4 (Q5, Table 1). There was no a clear result on whether they prefer an hour using the system from an hour tutorial session (Q3). 36% were in favor of the system, 39% in favor of the tutorial session and (surprisingly?) 25% introduced by themselves and checked a third choice saying that both are equally useful for learning.

The open questions revealed that the self-evaluation facility was more usable and useful than the open exercising facility, which needs technical improvements.

However, they suggested that more examples/exercises should be added, especially related to the NL to FOPC conversion, to cover more cases.

Q	QUESTIONS	ANSWERS (%)				
		1	2	3	4	5
2	How much did the system helped you to learn Logic?	0	3	29	39	29
4	Did you enjoy learning with the system?	0	3	21	61	11
5	Will you suggest the system to next year students?	0	0	14	32	54
6	Did you find the interface easy to use?	0	0	14	61	21

Table 1. Questionnaire Results (partial)

We also compared the exam results of the 2002-3 class, who did not use the system, and the 2003-4, who used it. The comparison is based on the questions of the exam paper related to the system topics (see Table 2). It is clear that there was an improvement to the class average mark as far as FOPC to CF conversion question is concerned, but a reduction as far as NL to FOPC conversion question is concerned. The first result is quite encouraging, because, apart from the improvement to the average mark, the given FOPC formula was more complex than that of previous year. This is also the reason for the second result, the reduction. The NL sentences were more complex than the ones in the previous year exam. That result is also consistent with one of the open questions findings: more examples/exercises related to NL to FOPC conversion should be added in the database, which is quite easy to do.

Question	Feb-2003 exam	Feb-2004 exam
NL-to-FOPC (15 marks)	10.7	7.53
FOPC-to-CF (15 marks)	13.1	13.5

Table 2. Exam Results

6. Conclusions

Contemporary education may not be based exclusively on class lectures, given the current capabilities of technology. Web-based educational systems are a class of systems suitable for a more personalized learning.

In this paper, we present a web-based education system to assist students in learning and tutors in teaching predicate logic as a knowledge representation and reasoning language.

The system leaves the initiative to the students of how they will use the system. The main characteristics of the system are: (a) a student can select on his/her own the next topic to deal with, (b) a student can be self-evaluated by specifying the difficulty level and the complexity of the exercises to try and (c) a student can try any FOPC to CF conversion through the automated open exercising facility. They make the system adaptable to student needs.

Recent developments in the area of web-based educational systems use AI techniques to achieve automated adaptation to user needs, resulting in adaptive web-based educational systems (see e.g. [3], [4], [7], [8]). Automated adaptation may be not always the best solution in a web-based educational system. However, this is a challenge and our future efforts are directed to incorporate intelligence to our system.

7. References

[1] Plato. http://www.utexas.edu/courses/plato/info.html (accessed February 2004).

[2] Logic Toolbox. http://philosophy.lander.edu/~jsaetti/Welcome.html (accessed February 2004).

[3] L. Lesta and K. Yacef, "An Intelligent Teaching Assistant System for Logic", Proceedings of the ITS-2002, Biarritz, France (June 2002) 119-128.

[4] A. Mitrovic and K. Hausler, "Porting SQL-Tutor to the Web", Proceedings of the ITS-2000 Workshop on Adaptive and Intelligent Web-based Education Systems, 37-44, 2000.

[5] I. Hatzilygeroudis and H. Reichgelt, "ACT-P: A Configurable Theorem Prover", Data & Knowledge Engineering 12 (1994), 277-296.

[6] LIJA. http://cube.misto.cz/lisp/ (accessed February 2004).

[7] K. Kabbasi and M. Virvou, "Using Web Services for Personalised Web-based Learning", Educational Technology & Society, 6(3), 2003, 61-71.

[8] K. A. Papanikolaou, M. Grigoriadou, H. Kornilakis and G. D. Magoulas, "Personalizing the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE", User-Modeling and User-Adapted Interaction 13(3), 2003, 213-267.