The Virtual Campus Prolog Learning Environment

Helmar Gust, Christoph Peylo, Claus Rollinger, Wilfried Teiken
Institute for Semantic Information Processing, Universität Osnabrück
Katharinenstr. 24, D-49069 Osnabrück
Email: {gust,cpeylo,rolling,wteiken}@cl-ki.uni-osnabrueck.de

June 4, 1999

Abstract

We will present a *Prolog Learning Environment* which has been implemented in course of the *Virtual Campus Project*[1]. The *Virtual Campus Prolog Learning Environment* provides all necessary information a learner needs during his first steps in programming language learning and offers error explanation capabilities to facilitate the learning process. The system encourages explorative and self directed learning. In this paper we concentrate on the way the *Virtual Campus Prolog Learning Environment* detects learner errors and explains them.

1 Introduction

The *Virtual Campus Project* is a joint project of the universities of Hannover, Osnabrück and Hildesheim launched in July 1997 as a three year research project. Its aim is to integrate Internet technologies into education and to develop lectures and environments for several courses accessible through the Internet. The Osnabrück part, an Internet-based intelligent *Prolog* learning environment, is developed at the Institute for Semantic Information Processing.

In the first part of the paper we will consider in which way the acquisition of complex cognitive tasks, like programming, may be supported by intelligent systems. In the second part we will introduce an intelligent Prolog learning environment, which has been developed in course of the *Virtual Campus Project*. The third part is on error detection capabilities of our system. Finally, we will give an outlook on further work.

In this paper we will distinguish between *Intelligent Tutor Systems* and *Intelligent Learning Environments*. We will use the term *Intelligent Tutor System* according to the definition of [5], [3] and [4]. Since *Intelligent Tutor Systems* are expected to be used *without* a human teacher [1] we consider an explicit tutoring component to be a main characteristic of them.

In contrast, we use the term *Intelligent Learning Environment* to denote an intelligent artificial environment which does not necessarily replace a human teacher.

[1]This Project is founded by the Ministry of Science and Culture of Lower Saxony

2 Learning to Program

The ability to learn how to put the solutions to problems into such explicit terms that a computer can "understand" them involves various skills and knowledge. Although the knowledge involved is comprehensive it may be related to specific domains. Thus, an intelligent system may offer a learner information and help – provided that the knowledge domains are sufficiently represented in the domain model of the system.

It is difficult to describe precisely which talents are required because programming may be regarded as an engineering task or, on the other hand, the creative aspects may be emphasised. Since the problem solving process heavily depends on the kind of abilities which are involved, a system which is to support a user during this process should be based on reliable assumptions concerning the involved talents. To enable a system to recognise a learner's plans and strategies the search space of the plan recognition task has to be limited. This often results in restrictive and obtrusive systems which force a learner to adopt himself to the solutions of the system or in systems which may be used in very limited domains.

Due to this problems we have chosen another strategy: analysis and explanation of (partial) solutions. This enables a learner to develop a programming strategy which suits him best and provide help and explanation to make him learn from his errors.

3 The VC Prolog-Learning-Environment

The present system is designed for supporting courses on logic programming and is accessible through the Internet (http://virtcampus.cl-ki.uni-osnabrueck.de). The Prolog environment may be used by students either to accomplish their assignments or to explore the realms of Prolog in a self directed way. It consists of three components:

Programming Environment: Mainly, the programming environment contains an editor and a control panel. The editor offers facilities for file access, compiling code and saving assignments. The control panel is the interface to the application process on the server to which it is connected through the Internet. It manages the user interactions with the server, e. g. executing programs and viewing the results.

***PLOT* component:** The *PLOT* component, which has been developed during a student's project, contains several components for syntactic and semantic analysis of the user's code, including a knowledge base of example solutions for given problems. We will go into details in the next section.

Lecture Notes In order to help the user to understand the basic structure of the logic programming domain, we provide Prolog lecture notes as a kind of semantic net which represents the basic concepts of programming and their relations to each other. The topology of the net forms an ontology which models the domain in a meronymic way. Thus, it provides a well-founded structure in which related domains may be integrated without violating the consistency of the model [2].

4 Error detection facilities

The ability to explain errors to the learner requires basic understanding of the learner's intention. Furthermore, a limited capability of program verification is required. Our system may provide a learner with detailed help to the extent he works on predefined assignments. To enable the system to analyse the learners solution, the system is provided with detailed information on the assignments. This includes knowledge on involved predicates and what the program is supposed to do. Furthermore, for each assignment exist example solutions for algorithms which could be applied to solve the posed problem. *PLOT* uses this knowledge to detect errors in a learner's code in different modules:

- the "syntax check" scrutinises the code for errors which would be detected by the compiler or interpreter.
- The "semantics check" requires example solutions to compare the behaviour of the learners code during program execution with the intended behaviour. The test is performed by a meta interpreter.
- The "structure check" applies default reasoning capabilities to detect lapses or slips which might result in semantic errors. This component relies on certain programming conventions which are taught in the lesson.

The errors which are detected in each component are passed to the "error handling component". This component contains a knowledge base with inference rules on errors and their relation to each other. Furthermore, it accounts for the certainty the single components may detect errors. The explanation of the detected errors is done by explanation based reasoning. Thus there is no static mapping of errors to error messages but the type, frequency and certainty of errors are drawn into account.

5 Further Work

Currently, we are working on a user modelling component which is based on data a user produces while interacting with the system. By applying Machine Learning and Data Mining techniques on the retrieved data will provide us with more information on the user's knowledge level and style of learning.

References

[1] J.R. Anderson, C.F. Boyle, A.T. Corbett, and M.W. Lewis. Cognitive Modeling and Intelligent Tutoring. In J. Clancey and E. Soloway, editors, *Artificial Intelligence and Learning Environments*, pages 7 – 51. MIT-Press, Cambridge, London, 1990. Reprinted from Artificial Intelligence: an International Journal, volume 42, number 1, 1990.

[2] N. Guarino. Formal Ontology and Information Systems. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 3 – 19. IOS Press, Amsterdam, Berlin, Oxford, Tokyo, Washington, 1998.

[3] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme*. Oldenbourg, München, Wien, 2 edition, 1997.

[4] G. Strube, B. Becker, C. Freska, U. Hahn, K. Opwis, and G. Palm, editors. *Wörterbuch der Kognitionswissenschaft*. Klett – Cotta, Stuttgart, 1996.

[5] B. P. Woolf. Ai in Education. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 434–444. John Wiley & Sons, Inc., 1992.