Journal of Industrial Technology 5 (1), 1995, 33-44

COMPUTER-AIDED INSTRUCTION PACKAGE FOR LEARNING PROLOG LANGUAGE

Mohamed Othman and Mohd. Hassan Selamat

Department of Computer Science
Universiti Pertanian Malaysia
43400 UPM Serdang Selangor
Malaysia

RINGKASAN: *Komputer telah lama digunakan di bidang pendidikan, yang kebanyakan hasilnya kurang memuaskan. Walau bagaimanapun, penyelidikan semasa di bidang kecerdasan buatan memberikan kesan yang positif kepada penggunaan dalam bidang pendidikan. Kewujudan idea Pembelajaran Cerdik Bantuan Komputer membantu melaksanakan satu sistem Pembelajaran bantuan Komputer yang mengajar bahasa Prolog. Ia ditulis menggunakan bahasa Prolog. Tujuan implementasi ini ialah untuk membina sistem Pembelajaran Cerdik Bantuan Komputer yang menyediakan persekitaran pembelajaran secara interaktif untuk pengaturcaraan bahasa Prolog. Semasa melaksanakan sistem, kecerdasan buatan memainkan peranan penting dalam pembangunan sistem. Walau bagaimanapun penyelidikan seterusnya diperlukan untuk mengatasi kekangan-kekangan sistem yang ada.*

ABSTRACT: Computers have been employed in the field of education for many years, often with disappointing results. However, the current research within the field of artificial intelligence is having a positive impact on educational applications. The existence of the Intelligent Computer-Aided Instruction ideas helped us during the implementation of a Computer-Aided Instruction system that teaches Prolog language. It is itself written from scratch using the Prolog language. The aim of this implementation is to be an initial step towards building an ICAI system which provides an interactive learning environment for Prolog programming. During the implementation of the system, artificial intelligence played a major role in the development of the system.

KEYWORDS: Computer-Aided Instruction (CAI); Intelligent Computer-Aided Instruction (ICAI); Programming in Logic (Prolog); lexical analysis; parser

INTRODUCTION

Computers, particularly microcomputers are now extensively use within the educational system. With decreasing cost and increasing capabilities, microcomputers have at last become accessible for use in the field of education. In addition, the rapid growth of CAI software development has posed even more challenges to the artificial intelligence research community.

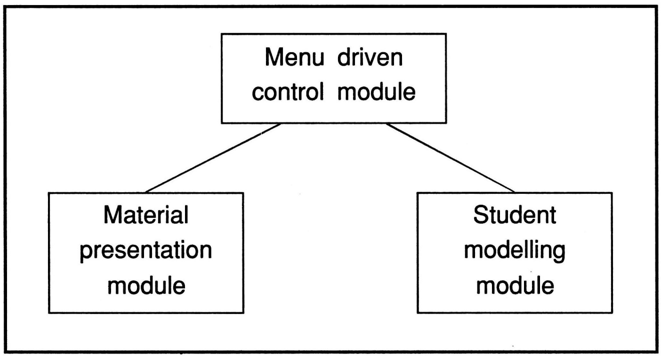


Figure 1. *Relationship among the modules*

Many CAI systems have gone beyond the drill-and-practice stage in attempting to provide what Seidel and Rubin (1977) called a reactive learning environment, in which a student is actively engaged with an instructional system. Such a CAI system must be capable of analyzing a wide range of students' responses. In other words, an ICAI system is an expert in a particular topic. In addition, an ICAI system generally contains a student model that presents both the student's knowledge and misconcepts, as well as a component containing information regarding teaching strategies. In most ICAI systems, (Barr *et al.,* 1981a, 1981b. Koffman and Blount, 1975; Sleeman and Brown, 1979, 1981; Sleeman and Smith, 1981) the course material is presented independently from both the student model and the instructional procedures or strategies.

This paper describes a CAI system for teaching Prolog which was developed at the Department of Computer Science, Universiti Pertanian Malaysia. This system is an aid to the students in the process of learning a Prolog program. The implementation is an initial step towards building an ICAI system for the teaching of Prolog on any personal computer (PC's) machine.

COMPONENTS OF CAI SYSTEM

The CAI system consists of three different component modules. Menu Driven Control (MDC) module, Material Presentation (MPM) module and Student Modelling (SMM) module. These three modules correspond to the feature of an ICAI system like the Tutoring Strategies module, Expertise module and Student Modelling module, respectively and their relationship is shown in Figure 1 (Sleeman and Brown, 1981).

The MDC module makes use of three menus to interact with the user and decides on the progress of the tutorial process. It controls the MPM module in presenting appropriate material at any time based on the request from the student and the result from the SMM module. The control will pass all the necessary information to the SMM module and activate the student modelling process. The control will always pass back to the top level menu after each sub-task is completed.

The expertise knowledge of this system is the Prolog language and the materials are mostly taken from the standard text written by Clockin and Mellish, (1981), and Mohamed Othman, (1989). In the MPM module, the material presentation routine uses the material from a file to give instructions to the student. The SMM consists of a problem generator and a problem solver. The problem generator is capable of providing some meaningful problems selected from the file. A vital information link, in the form of indicators, exists between the problem generator and problem solver. This link provides the system with the information for the solution to the problem, and allows the problem solver to concentrate on the solution.

Functions of CAI system

A new student can use the material sequentially to learn the Prolog Language. He can control the speed of the tutorial process at any point in a session. The system can detect the mistakes of the user and provide remedial material to ensure the user understands the material presented to him. Furthermore, the system can function as a consultation program. If the user encounters a problem with one of the built-in predicates in Prolog, he can instruct the system to provide information about a directory of all predicates. The student can refer to this directory to find out the exact spelling of the predicates.

During the tutoring process, the system will provide the student with some exercise to test his understanding of the material. During the problem solving process, the student is constantly being given feedback as to the correctness of his solution. If his response contains syntactic or semantic error, the system will point this out and offer helpful suggestions.

The system is constantly evaluating the student's performance through updating the history file. This is necessary because his achievement determines the problems that will be given to him. It should be stressed at this point that the system can only solve the primitive tasks such as multiple choice questions, query statement and simple Prolog program.

User friendliness

The system is very user friendly. It provides easy-to-follow instructions. The student needs only to give simple answers to the system to control the flow of the program especially when he makes mistakes during interaction with the system. For example, when the student chooses an option which is not available, the system will inform the student about the error and allow him to re-enter his choice again.

The system design

The system is written using Prolog Language. The concept of structured and modular programming methodology was introduced during implementation of the program. It can be divided into two main modules, namely lexical analyzer and parser.

The lexical analyzer is used for analyzing English sentences. It is capable of processing Prolog features such as the "." at the end of every clause because "." is considered to be part of a word in the English sentence. Also after the operators ":-", a space must be entered before any other clause because "-" is treated as part of an English word.

Similarly, the parser is capable of parsing Prolog commands and clauses starting with the symbol "?-". It also generates appropriate error indicators to the SMM module for action to be taken. Both modules are used by SMM to analyze the syntax of any response given by the user.

```
                    PROLOG TUTOR SYSTEM

                    Welcome to Prolog tutoring system

In order for the system to keep track of your progress, please enter
your PPN when prompted by the system

Is this the first time you are using this system (y/n) ? n

Please enter your PPN number?
>> 12345.

You did not use this PPN: 12345 before, or it is possible your record
has been deleted because you did not use this system for the past
two months.

Please enter your new PPN number?.
>> 12345.
```

Figure 2. *Student's background*

The system can be converted to teach different subject matters by merely changing the contents of the expert domain. This is an interesting feature in the ICAI system. For example, EMYCIN is an expert system (Barr *et al.,* 1981a, 1981b and Hayes Roth, *et al.,* 1983) which can be used in a number of different problem domains.

The system is capable of handling all the requirements of a tutoring system. It can be easily expanded by adding new problem types without changing the system's problem solving capability.

SYSTEM OPERATIONS

This section presents a general description of the system operations. It consists of three main modules as follows:-

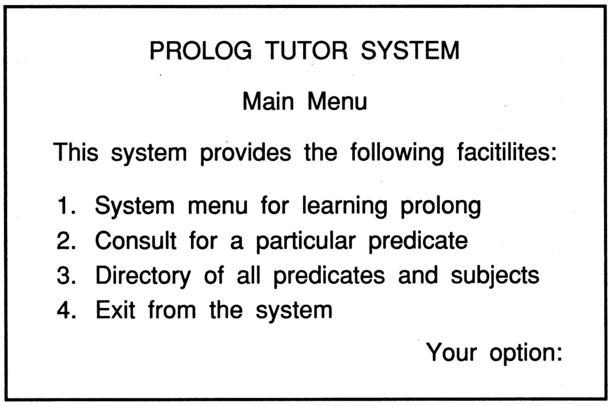
Menu Driven Control Module

This module specifies how the system presents the material to the student and performs important housekeeping work for the system.

Student's Background

At first, the system loads the student's history file so that the past history of the student can be used to monitor his/her progress and select the problems for the student. It needs to load in the history for all students who have used this system before. Prolog only allows the user file to open simultaneously as input and output file. Otherwise, the system will not be able to perform the maintenance process on the file.

The systems then displays the introduction to the systems, gives detail on what the system can do and how to start it. The system will then ask the user to enter his identity as shown in Figure 2. Each user is supposed to use a unique PPN number. If the system notes a new user using others' PPN, the system will reject the user and ask to re-register himself. The same procedure happens if the PPN given by the user who claims to have used the system before does not exist in the history file. This is to ensure the actual background of each individual student and the accuracy of the modelling process. After the system has confirmed the status of the user, a predicate current (PPN, Level, Num, Date) is used to indicate the current section of the material of which the user is studying.



```
                 PROLOG TUTOR SYSTEM

                      Main Menu

  This system provides the following facitilites:

  1. System menu for learning prolong
  2. Consult for a particular predicate
  3. Directory of all predicates and subjects
  4. Exit from the system
                                      Your option:
```

Figure 3. *Main menu*

For a new user, Level and Num will be initialized to zero. For a user who has used this system before the values for Level and Num will be taken from his history record to indicate the point where he last stopped. After the student's background has been established, the system enters its top level control by printing the main menu as shown in Figure 3.

The control is then passed to the appropriate sub-control of the MPM on the option chosen by the user. It has to pass back to this top level control at the end of the sub-task for a proper exit from the system because the system needs to perform the maintenance of the history file before terminating the process.

Housekeeping Task

Whenever a user decides to exit from the system, the system has to perform the housekeeping work. At first, it accepts the current date from the Personal Computer System followed by the Level and Num values and then updates the history of the student.

After that, the system will look for the history records of inactive students for the past two months and delete their records. All predicates that were associated with them will also be deleted. The system terminates by displaying a good bye message.

Material Presentation Module

The MPM obtains the control from the top level. It follows the option chosen by the user to activate the second level control for the desired path.

Sequential Presentation

When a user requests to follow the course sequentially, this segment of the program will be used to control the presentation of the material. If this is the first time the user is using the system, the system will start at the beginning of the material file, otherwise the system will print the second menu. Next, the user has the choice of several options such as to start either from the beginning of the course, from the point where he last stopped or from any point to be specified by him. All these options are shown in Figure 4.

```
                 PROLOG TUTOR SYSTEM
                   Sequential Menu
        Please enter one of the following options.
  1. Start from the beginning of the course
  2. Start from the point you stop last time
  3. Start from the point to be specified by yourself
  4. To look at the directory
  5. Return to previous menu
                                          Your option:
```

Figure 4. *The Sequential menu*

To start from the beginning of the course the system simply changes the value of Level and Num of the current predicate to zero and calls the common routine to process the material from the beginning.

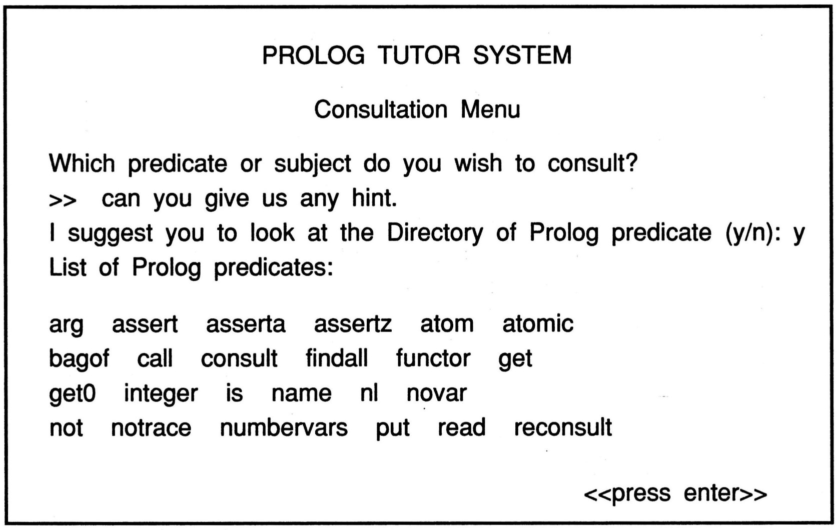
To start from the point where the user last stopped, the system will use the values of Level and Num from the current predicate to by-pass the material before these values. The common routine is then called to start from that point.

In order to start from the point specified by the user, the system displays the third menu for the user. The user can specify the name of the predicate or subject he wishes to start from. The name given by the user must exist in the directory, otherwise the system will reject it and allow the user to input another name, or return to the previous menu, or look at the directory of predicates. When the name is found, the corresponding Level and Num values will be used as the entry point, and the user again has the three options described above.

In sequential presentation, the user has the option to revise the subjects after he has completed the whole course.

Consultation Presentation

The user can consult the system for the information on predicates by entering its name. The directory is again used to validate the user's input clause. The user can specify the entry point to the course. After the desired entry point is established, the same presentation routine is called to present the material. The control is returned from that routine when it is completed. The user is allowed to consult other predicates, otherwise this routine will give the control back to the top level menu.



```
                     PROLOG TUTOR SYSTEM

                      Consultation Menu

Which predicate or subject do you wish to consult?
>>  can you give us any hint.
I suggest you to look at the Directory of Prolog predicate (y/n): y
List of Prolog predicates:

arg  assert  asserta  assertz  atom  atomic
bagof  call  consult  findall  functor  get
get0  integer  is  name  nl  novar
not  notrace  numbervars  put  read  reconsult

                                              <<press enter>>
```

Figure 5. *Consultation menu*

During the consultation, the user can display all the predicates available in the directory as shown in Figure 5. It can be made only at the top level control or when the choice is given by the system. In case of any request for non-existence predicate, the user-friendly system will automatically display the directory to enable the user to make a correct choice.

Presentation Routine

The common routine is used to read the file sequentially and process each line at a time starting from the entry point. It makes use of the backtracking feature provided by Prolog language to process the material which includes lines of text, indicators and others.

The routine will display the line of text on the screen. The indicators are read from the file and the processing takes place according to the following description.

<end_of_file_marker> when the routine reads this marker, control will return to the calling routine.

<end(Level,Num,_)> at the end of one section, the routine updates the student's history to indicate that he has gone through this section and increases the page counter. At this point, the user has four choices.

1. Go to previous page. The routine allows the user to go back one page of material at a time until the first page of this section.

2. Continue to the next section.

3. Exit from the system. The routine will call the end of session routine to perform the housekeeping work and exit from the system.

4. Return to the top level menu.

<sub(Level,Num,_)> enter a new section. This routine updates the current pointer to indicate that the user is now studying this section.

<check(Level,Num,Set)> at this point, the control will be passed to the SMM. The exercise identified by Level,Num and Set will be generated for the user.

<page(Level,Num,Count)> a new page of material is about to be printed to the user.

Student Modelling Module

The SMM consists of the Problem Generation routine and the Problem Solving routine. Basically, the first routine is activated to select the appropriate problem for the user and then the control is passed to the Problem Solving routine to model the student's response.

Problem Generation Routine

This routine uses the section indicators to search for the problem in the file for the corresponding problem sets. The problem to be presented to the user is selected by the following algorithm:

1. The number of problems in the problem set is obtained from the no-of-prob(No) indicator, which is supposed to be the first line in the problem set.

2. A set of accumulators is created to store the score for each problem when the pre-condition in the problem set is processed. The number of accumulators is the number obtained in Step 1.

3. A set of pre-conditions is then read from the file. If the pre-condition can be satisfied, the problems indicated by the list in the pre-condition will score a point. The points are added to the corresponding accumulator.

4. Finally, the problem with the most points will be selected and passed to the Problem Solving routine.

5. End.

Problem Solving Routine

This routine will process the material. When a line of problem description is read, the routine will display it on the screen and output it to a temporary file in case the user needs the problem definition later on. A special indicator encountered is processed as below:

<answer(Ans-list, Remedial)> this indicates that the problem is either a multiple choice question or the problem requires only a simple answer and the possible answers are given in the Ans-list. The routine will ask the user to enter his response and check to see if it is one of the possible answers. If the response is correct, the routine will continue with the material until the end of the problem. Otherwise, the remedial material indicated by Remedial will be printed to help the user. If the user still cannot provide the correct answer, the routine will print the possible answers on the screen.

In some cases, the problem given to the user requires more than one answer. The routine has to make sure that the user does not give the same answer more than once. To ensure this, the routine stores all the previous answers given by the user. Any duplicate answer will be rejected. When the following lines are read by the routine, they are asserted into the Prolog database and they will be used to check the user responses.

<info(Type,Predicate,No-of-arg,State-Inst,Operation)>

and

<aug(Type1,Predicatel,No1,Type2,Predicate2,No2)>

The Lexical Analyzer will be called to analyze the clause input and the parser will be used to check the syntax of the input clause. Any error encountered will be recorded. The Problem Solving routine will inform the user about the error and allow him to re-input the clause. When the clause is free of syntax error, production rules are used to check the clause against the stored information such as predicates, number of arguments, state of instantiation and operations. Again, errors encountered will be reported and the user is allowed to input another clause.

Production Routine

There are two sets of production rules. The first set of production rules is used to prepare all the necessary information to be used by the second set. This information is obtained from the user's input clause, which has to be free from syntax error, and asserted into the Prolog database. The syntax is stated as follows:

<clause(Type,Predicate,No-of-arg,State-inst,Operation)>

This syntax is identical to the predicate info (-,-,-,-,-) in file so that the production routine can check the matching of the arguments easily.

The second set of production rules is used to check the user's input against the stored information. The elements of the clause are checked in the following order:

1. The predicates of the clause are checked first. Two types of errors can occur: mis-spelling and missing predicate.

2. The routine then compares the number of arguments of each predicate.

3. The actual arguments are then compared. Connection between arguments in different predicates is checked.

4. Finally, the routine checks the operators of the clause.

5. End.

If an error occurs at any stage, the routine will reject the user's response and the subsequent test is not carried out.

CONCLUSIONS

This system presented in this paper could be used as a tool for teaching a Prolog programming language. It has achieved our aim of setting the system as an initial step towards an ICAI system although it still has some limitations as to what can be expected from a truly ICAI system. The system can be extended provided we can overcome the problems of domain expert representation, modelling the arithmetic calculation, intelligence reasoning and semantic.

REFERENCES

Barr, A. and Feigunbaum, E., (1981a). In: *The Handbook of Artificial Intelligence,* Vol. 1, Addison-Wesley, New York.

Barr, A. and Feigunbaum, E., (1981b). In: *The Handbook of Artificial Intelligence,* Vol. 2, Addison-Wesley, New York.

Clockin, W.F. and Mellish, C.S., (1981). In: *Programming in Prolog,* Springer-Verlag, Heidelberg.

Hayes-Roth, F., Waterman, D.A. and Lenat, D.B., (1983). In: *Building Expert System,* Addison-Wesley, New York.

Koffman, E.B. and Blount, S.E., (1975). Artificial intelligence and automatic programming in CAI. *Artificial Intelligence J.,* **6,** 215-234.

Mohamed bin Othman, (1989). Logic and Prolog database system. *Teknologi,* **14**, 41-48.

Seidel, R.J. and Rubin, M., (1977). In: *Computer and Communication: Implication in Education.* Academic Press, New York.

Sleeman, D.H. and Brown, J.S., (1979). Intelligence tutoring system. *Special Issue of the International Journal of Man-machine Studies,* **11(1).**

Sleeman, D.H. and Brown, J.S., (1981). In: *Intelligent Tutoring System,* Academic Press, New York.

Sleeman, D.H. and Smith, M.J., (1981). Modeling student's problem solving. *Artificial Intelligence J.,* **16**, 171-188.

Wirth, N., (1976). In: *Algorithms + Data Structures = Programs,* Prentice-Hall, New Jersey.