The design of logic programming learning system with Prolog

The design of logic programming learning system with Prolog

Hiroyoshi BABA, Toshikazu KUROSHIMA and Ichiro SUGIOKA

Abstract

The purpose of this research is to design a logic programming learning system as a type of computer-assisted instruction by adopting Prolog language. This system presents practical programming problems to students so that they can understand the concept of logic programming in a process of solving practical problems. This system has the feature that realizes multi-windows on screen that attracts students' interests for learning, and has the help function that corresponds to requests from students. By using the function, students are able to control the courseware in this system of their own accord.

1. Introduction

The concept of logic programming comes from a result knowledge and understanding both with regards to programming theory and logic in general. Strictly speaking, logic programming consists of a set of truth clauses and their logical reduction, but practically speaking, it is realized by two programming methods —— database and recursive functions[1]. More concretely, logic programming is related to declarative programming based on static notation instread of procedural programming based on dynamic notation. Prolog language, that is derived from predicate logic, is available to execute logic programming in reality on a computer.

A computer operates various coursewares depending on the degrees of the students' understanding. Namely, it has some aspects that superior to human beings in education because it enables to show them the optimal learning steps in a courseware[2]. This tutoring system is designed with Prolog and it has two control modes —— learner control mode and author mode. The students can change the mode in this system when they need some helps in learning.

2. Logic programming

2.1 Definitions

There are two kinds of definitions of logic programming[3] :

1). Logic programming is to write a program using "logic programming language" based on pre-

dicate logic form.

2) Logic programming is to formalize a problem using logic forms and to produse a program under converting from the formalization.

The difference between their ways of thinking is that the first definitions stresses on the language itself while the second one stresses on the processes in making a program. In other words, the first logic program may include nonlogical expression and the second logic program may be written in a procedural language such as Pascal or FORTRAN. The common concept of logic programming is how describes a problem logically and how interprets such an logical expression procedurally.

On the other hand, Prolog is the language that can realize logic programming on a real computer. Also, it is the fact that Prolog language has some procedural expressions and it cannot cover all concepts of logic programming. But Prolog is the first programming language as the result of many years of the research of logic programming.

2.2 Database programming

Logic database consists of a set of facts and their rules. As an example, "Family" database is discussed here to describe database programming. When we define the relationship between 'father' and 'child' as one of facts, just define the following predicate:

father(Father, Child).

Instead of the above expression, when we define the relationship by using rules and suppose the relationship 'parent' and 'male' have already defined as facts, define a new rule as follows:

father(Dad, Child) : −parent(Dad, Child), male(Dad).

It means that if Dad is a parent of Child and Dad is a male, then Dad is a father of Child. From the viewpoint of logic, it is not significant that the relationship is defined whether as a fact or as a rule.

As an application of database programming, to arrange the structure of data is significant. By structurizing of data, it is possible to describe data which have more semantics and it is possible to rules more abstractly, in other words, the property of module in programming comes to be more important. When all of data are structurized well, a program has clearer perspective especially in the condition of programming of complicated problem.

2.3 Recursive programming

To apply mathematical functions in programming, it is necessary to have a kind of infinite structure in the program. Recursive data type can realize such an infinite structure and also can realize more elegant program as the result.

The following procedure is recursive because the last line is a call to the procedure itself[4]. Thus, any recusive procedure includes at least one of each of the following components:

(1) A nonrecursive clause where the recursion stops;

(2) A recursive rule. In the body of this rule, the first subgoals generate new argument values. Then follows a recursive subgoal utilizing the new argument values.

2.4 Prolog

Prolog programming consists of a set of a following sentence:

A : −B1, B2, ..., Bn.

"A" is the head of a sentence and "B1, B2, ..., Bn" are called the body of it. The symbol, ":−" is the same meaning as "if". This sentence can be classified three patterns as follows:

1). Facts. "A." (A is fact(true).)

2). Queries to the database. "? :− B1, B2, ..., Bn." (B1, B2, ..., Bn are fact(trues)?)

3). Rules. "A :−B1, B2, ..., Bn." (If B1, B2, ..., Bn are facts(trues), then A is fact(trues).)

These three patterns are called Horn clauses that are restricted predicate logic forms based on first order predicate logic.

3. Computer-assisted instruction

3.1 Learner control mode

As one of ways of computer-assisted instruction, learner control mode is a kind of student-driven system that students can learn subjectively, that is, the efficiency of learning is gained in the interaction between the students and computers. To do this, the following conditions are required:

1). To implement easily of the pregress of courseware, the level of contents, and the choice of the direction of learning which the system prepares depending on the intention of the students' side.

2). To understand the requests and questions from the students in a natural language or in other ways of communication.

3.2 Author control mode

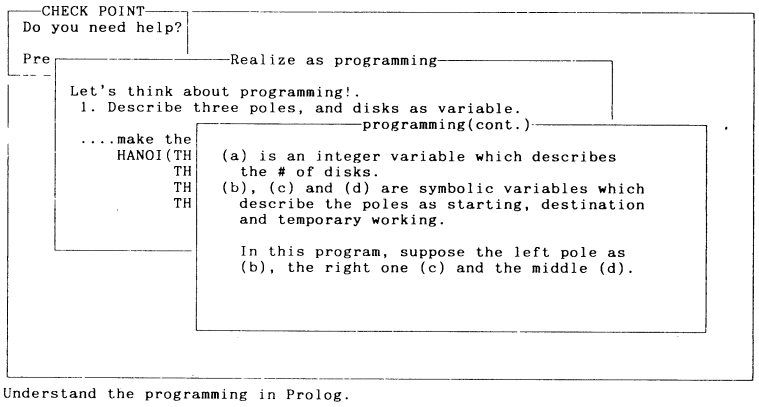
This mode is also called program-driven mode which means all of processes and stages are controlled in the courseware the author designed. So the students traverse the courseware under the author's guide. But it is hard to reflect students' desire in this control mode because all of the steps in the courseware have already been implemented.

4. Tutoring system

4.1 system environment

IBM PC true compatible computer (comptalk, Inc.) is used as Hardware environment, and Turbo Prolog which is a compiler of Prolog language and Turbo Prolog Toolbox which can enhance Turbo Prolog are used as software environment under MS-DOS Ver.3.3, an operating system.

This tutoring system has the feature that appears multi-windows on screen. Screen 1 shows an example of multi-window and that of a status line.



Screen. 1 An example of multi-windows.

4.2 Courseware

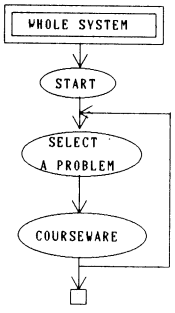


Fig. 1 A diagram of the whole system.

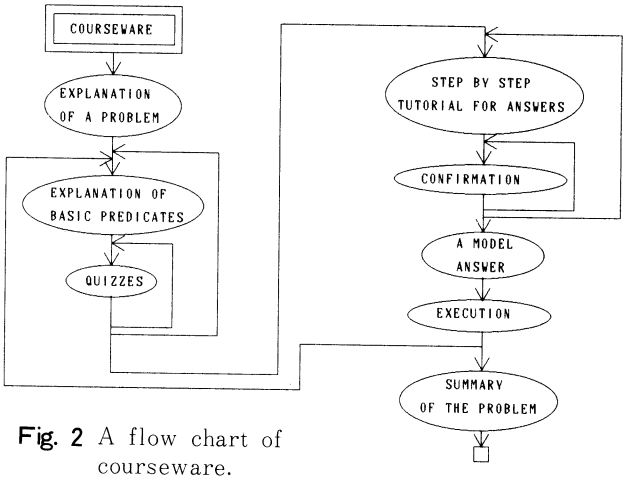


Fig. 2 A flow chart of courseware.

This tutoring system presents concrete programming problems to the students and the student will understand logic programming in a process how to solve practical programming problems.

Figure 1 shows the whole construction of the tutoring system. The diagram shows that students select a specific problem and the corresponding courseware opens. Once the student finishes a courseware, one can feedback to the stage that selects another problem depending on the student's intention. The tutoring system is closed if the student desires to quit to learn.

Figure 2 shows the flow chart of a courseware. Each step is realized on a sccreen by using a multi-window.

There are two stages in each courseware —— a fundamental part and a applicated part. The left side of the chart corresponds the fundamental part and the right side does the applicated one.

There are three steps in a fundamental part —— 'explanation of a problem', 'explanation of basic predicates' and 'quizzes'.

At the step 'explanation of a problem', the student makes sense an outline or rules of the specific problem one selected. The following step 'explanation of basic predicates' makes the student understand the specific some of predicates in Prolog programming that are required to solve the practical problem. And the student is able to deepen one's knowledge by answering quizzes operated the step 'quizzes'. The quizzes have a shape of multi-choices, namely, the student choose the proper choices by pressing the Return key seeing the menu on a screen.

Before finishing a fundamental part, some of feedbacks can be operated depending on the level of the students' understanding; the same quizzes are repeated until the student can find correct answer(s), otherwise, the student returns back to the step 'explanation of basic predicates' in order to review the correspond basic predicates.

There are five steps in an applicated part —— 'step by step tutorial for answers', 'confirmation', 'a model answer', 'execution' and 'summary of the problem'.

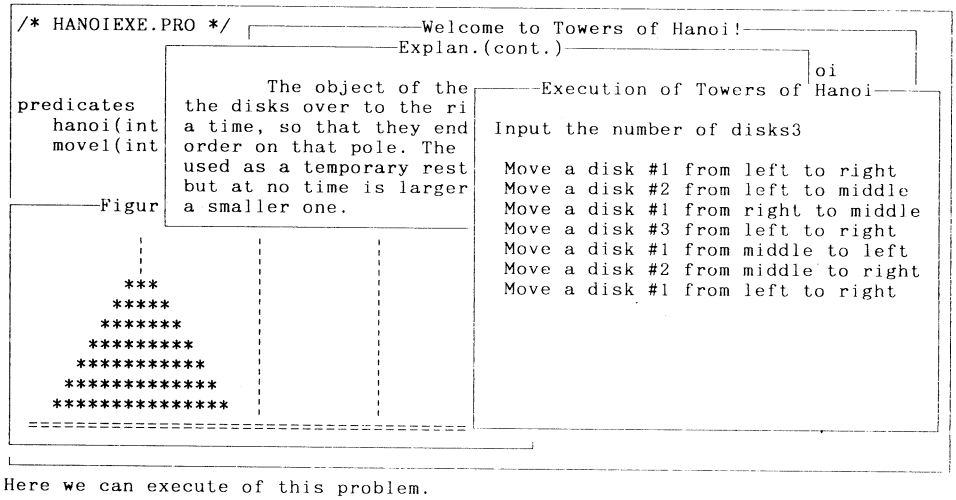
1) 'Step by step tutorial for answers'

After understanding of basic predicates, the student is led to the answer step-by-step under a top-down method.

2) 'Confirmation'

The student can confirm the approaches that a courseware presents by reiterating the same multi-windows or by using the help function on this stage. Like the last part of a fundamental part of a courseware, some of feedbacks will be operated between the stages 'step by step tutorial for answers' and 'confirmation'.

3) 'A model answer'



Screen. 2 An example of the stage of 'Execution'

The stage that the student can see through the whole programming list of the problem.

4) 'Execution'

The student executes the program of the specific problem one selected.

5) 'Summary of the problem'

As a summary of solving the problem, this stage presents a brief background of the problem, some supplements from a view point of logic programming, or formulas as the standard solving methods.

Before entering the final stage 'summary of the problem', if there are other solution of the problem, it is possible to occur feedback returns to the beginning of a fundamental part of the courseware. Screen 2 shows an example of the stage of 'Execution'.

4.3 File structure

There are ten different files that consist the tutoring system —— 'main.pro', 'menu.pro', 'help-.pro', 'compred.pro', 'dangcave.pro', 'nqueens.pro', 'hanoi.pro', 'caveexe.pro', 'queenexe.pro' and hanoiexe.pro'. The function of each file is described as follows:

1) 'main.pro'

This file contains a main menu of the system and sub menus for the problems which have plural solutions.

2) 'menu.pro'

This file is the software tool to realize a menu window on a screen.

3) 'compred.pro'

This file contains the explanations of basic predicates and quizzes for understanding their predicates.

4) 'help.pro'

This file organizes the help function of the tutoring system. When the student loses a way in the courseware toward the complete understanding of the problem, this file is called by the student's request. The file 'help.pro' provides a list of items in the problem using a menu window with the student, and make the student choose the topic that one should review. The student can open the file 'help.pro' by selecting the HELP in the menu that appears several times in each courseware.

5) 'dangcave.pro'

6) 'nqueens.pro'

7) 'hanoi.pro'

These files are the coursewares for learning the practical problems such as 'A DANGEROUS CAVE', 'N QUEENS PROBLEM' and 'TOWERS OF HANOI', respectively.

8) 'caveexe.pro'

9) 'queenexe.pro'

10) 'hanoiexe.pro'

These are the files that contain source lists for executing the problem 'A DANGEROUS CAVE', 'N QUEENS PROBLEM' and 'TOWERS OF HANOI'. These file are also compiled when the whole system is compiled at the same time because Turbo Prolog is a compiler, not an interpreter like other Prolog systems.

Figure 3 shows the relationship among all of files for the system by a diagraph chart. Arrows express which file includes others, namely, 'main.pro' includes six other files —— 'menu.pro', 'help-.pro', 'compred.pro', 'dangcave.pro', 'nqueens.pro' and 'hanoi.pro'. Also each file contains a course-

ware for the specific problem includes the file has its programming source list such that 'caveexe.pro' is included in 'dangcave.pro', etc.

Figure 4 shows the whole structure of the tutoring system. A student picks up one problem of the three, and if the problem has two different solutions, one chooses one of them by sub menu. Each 'problem' has its own courseware and it accesses a set of 'basic predicates' for fundamental parts of courseware, 'programming methods' for applicated parts, general explanations' for the summary of the problem.

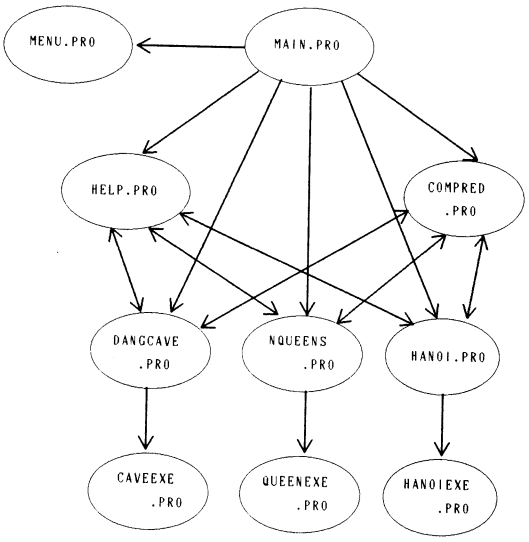


Fig. 3 A digraph of the relationship among files.

5. Consideration

On-screen multi-windows in this tutoring system generate two advantages: the one is that windows attract the student to the system, namely, they don't lose their interests. The other is that windows can indicate the end of each step in the specific courseware firmly by driving multi-windows one after another on a screen. It also helps the student to check until which step one reaches.

This tutoring system provides the student a help up to the student's request. By using this function, the student can control the courseware,

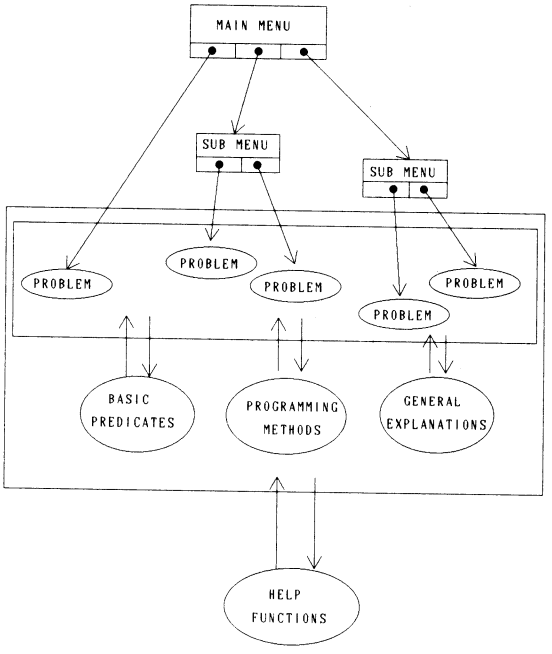


Fig. 4 The whole structure of system.

therfore, this tutoring system can be regarded as one of type of student-driven system.

But the concept of logic programming itself is still under researching and it is restricted to the use of Prolog only as the logic programming language in this system. It is also a firm fact that Prolog has some nonlogical functions and it cannot cover all the categories in logic.

This system has a help function for the student who need a help, but the function is operated by menu window, not by a true interactive condition such as natural languages. Therefore, this tutoring system does not understand the student's true request with or without one's ambiguous intention. To fulfill the purpose of designing true intellectual tutoring system, it may be required to arrange man-machine interactive communication feature in some of natural languages.

6. Conclusion

This tutoring system has an important aim to make students learn logic programming, not only to make them understand Prolog language, so that the student who uses this system can confirm the essential thought of Prolog and the reason why Prolog language should be available to realize logic programs on a computer. The method of this system which provides the student practical programming problems is able to be called 'top down method' so that the student can recognize the effects of one's learning with grasping the concrete purpose —— to solve the specific program. Also, the student is able to master some practical required techniques for logic programming such as database programming and recursive programming.

References

1) Leon Sterling, Ehud Shapiro 著，松田利夫訳：Prolog の技芸，共立出版
2) 芦葉浪久著：CAI コースウェア作成技法，東京書籍
3) 黒川利明著：Prolog のソフトウェア作法，岩波書店
4) John Malpas：Prolog – A relational language and its applications, Prentice Hall