8-16-1996

A Knowledge-Based System to Assist Teaching for Large Classes

Joshua Poh-Onn Fan
Department of Business Systems, University of Wollongong

Tina Kwai-Lan Mak
Department of Accounting and Finance, University of Wollogong

Li-Yen Shue
Dsepartment of Business, University of Wollongong

A Knowledge-Based System to Assist Teaching for Large Classes

Joshua Poh-Onn Fan
Department of Business Systems, University of Wollongong,
NSW, Australia.

Tina Kwai-Lan Mak
Department of Accounting and Finance, University of Wollongong,
NSW, Australia.

Li-Yen Shue
Department of Business Systems, University of Wollongong,
NSW, Australia.

Introduction

The problems with the conventional approach to large classroom teaching, which rely mostly on textbooks and the questions of review exercises at the end of each chapter, are well known. Apart from the fact that face-to-face consultation time can never be sufficient between students and lecturers, many lecturers cannot afford the time to provide detailed remarks for each question. In many instances, only an overall remark is given at the end of an assignment, which may not provide any direct assistance to students in problem solving. For most wrongly answered questions, it is up to the students to find out the correct ways of solving them. A consequence that arises is the fact that some students tend to relate solution approaches with particular exercise problems, and find it difficult to extend the same solution approach and the logic of problem analysis to problems of different settings (Nachouki and Gouarderes, 1994). Another well known problem with large classes is the tendency for lazy students to plagiarize and hence reduce the learning effectiveness. In the past, the only way to reduce these problems is through providing more teaching assistants, which is generally very costly and has proven ineffective in many cases (Flechsig, 1989; King and McAulay, 1992). Recently, the promotion of the Computer Assisted Instruction (CAI) has targeted these areas.

Most CAI packages (Bourne, 1990; MacKnight and Balagopalan, 1989; Pogue, 1985) were designed to store pre-prepared questions in a database. These questions will then be retrieved either randomly or according to certain index when needed. This approach may prove to be valuable for the kinds of topics whose contents do not require updating on a regular basis. However, it is apparent one may encounter problems in applying current CAI to teaching in a university environment. The problems are two-fold. Firstly, it is well known that the development of large pools of questions for a topic is very costly and time-consuming, it will be even so for university courses. Secondly, it is most likely that the question developer is not the lecturer himself/herself. This second problem highlights the lack of control by the lecturer over the content and the nature of questions to be given to students (Hannafin and Peck, 1988; Lockard, 1992). This lack of control of content may lead to the fact that the pre-prepared questions do not match the specific contents of a particular course (Nachouki and Gouarderes, 1994). In addition, a common criticism of CAI is the fact that most systems are designed to allow the marking to be carried out through matching users' answers with stored answers. Hence, the result for a particular question is either correct or wrong, and it is only the final answer that gets evaluated; the intermediate steps will not be awarded any partial credits (Hannafin and Peck, 1988; Lockard, 1992) unless each intermediate step is designed as a separate question. This may present a serious problem for the marking practice in a university environment. Also most CAI packages are not designed to provide feedback to guide students in problem solving (Bourne, 1990; Cook and Kazlauskas, 1993; MacKnight and Balagopalan, 1989; Milheim, 1993; Pogue, 1985), which can significantly reduce the effect of "instruction".

In order to overcome these problems, we incorporate the knowledge-based system design approach in the development of the Knowledge-Based Computer Instruction System (KBCIS), which was designed as an effective teaching-learning and examinations tool for large Accounting classes.

The Knowledge-Based Computer Instruction System

The goals of this system are to emulate and extend the role of lecturers in providing consultation service, and to provide a platform for efficient development of examination questions with flexible marking schemes. To achieve these, the System consists of an Agenda Scheduler, a Logfile, a Diagnostic Subsystem, an Assessment Subsystem, a Data Base, and a Knowledge-Base module.

The Agenda Scheduler serves as the main controller for initiating the execution of other subsystems, and the front end module to interact with both lecturers and students. Lecturers use a set of built-in macro commands to structure contents and exercises for a topic, and set up examination questions along with a specified marking scheme. Students use the Agenda Scheduler to retrieve course contents, exercise questions, and input their working answers. The Logfile is to chronologically record usage history of each student including answers to questions, which will be used by other subsystems for monitoring purpose. The Diagnostic Subsystem retrieves individual records from the Logfile to determine the level of assistance the system should provide to a student. The Assessment Subsystem is to implement a given marking scheme for an exercise or examination. The database subsystem stores representative questions, solution procedures, hints, guidance, and tutorials, which can be retrieved by other subsystems.

Knowledge-Base Module

The main focus of this system is the development of the Knowledge-Base module. The Knowledge-Base module is to take the solution procedures of a given question from the database and translates it into sets of problem solving rules with the final solution and intermediate solutions treated as the final goal and intermediate subgoals. It then applies the backward chaining approach (Durkin, 1994; Luger and Stubblefield, 1993; Medsker and Liebowitz, 1994) to seek the solutions of these goals. This approach takes the expected result of the final solution variable as the final goal, the expected result of the second last variable as its subgoal, and so forth. In this way, the correct answer of a question can always be derived as long as the values of variables for each subgoal are known. This is implemented through the following recursive algorithm:

```
Search for the goal of the question
While contained unknown subgoal
    Search for the unknown subgoal
    If no more unknown subgoal
        Carry out prescribed instruction to obtain subgoal
    EndIf
EndWhile
```

Problem Generation

Different settings of a given question can be generated by varying the values of a set of pre-determined variables, which are specifically designated by the lecturers. The values of these variables, both numerical and symbolic, can vary according to the given ranges and procedures. Each time a question is executed, the system will generate a new set of values for these variables, and students will be presented with a "new" problem to solve. Once the answers from students are entered, the Agenda Scheduler will then pass the control to the Knowledge-Base module to translate the solution knowledge of that question into decision rules with goal and subgoals, and find answers through the backward chaining process. Markings can then be made according to the assessment scheme specified in Assessment Subsystem. As a result, any given problem whose solution knowledge can be expressed in proper procedures can be used by the system to generate problems of different settings, and they are all of the same nature and requiring the same solution approach.

With this approach, the solution procedures for different settings of a given problem will always be the same, despite the facts that the values for some variables may be different for different settings. This approach can avoid the expensive part of storing a vast number of questions, which is a common practice of most CAI systems. The following examples demonstrates the variety of questions in Accounting that can be handled by this system.

Examples

The first question represents a straight forward calculation type of problems, which is to calculate the depreciation value using straight line method. The second question represents a typical multi-stage problem, where depending on a given answer the next subsequent question can be initiated.

Question 1 :

{AnyName} Ltd uses a Replacement Price Accounting System, and depreciates long term assets at 20% (straight line) per annum. The replacement cost of a plant in new condition was ${Amount1} at January 1995 and increased by ${Increased-Amount} at January 1996. Its replacement cost in its used condition at January 1996 is ${Replace_Amount}. What is the amount of depreciation on plant for the period ?

In this question, the variables inside bracket {} are the ones whose values are subject to change. The variable AnyName can have its value chosen randomly from the list provided by the lecturer. Assuming the value of Amount1 varies from $100,000 to $900,000, it can be expressed as ((random MOD 9 + 1)*100000). The cost of a new plant {Amount2} for 1996 can be calculated as (Amount1 + Increased_Amount), where the Increased-Amount is assumed to vary in the range ((random MOD 9 + 1)*10000). The Replace_Amount is calculated as (Amount1 - Increased_Amount), and the final answer is ((Amount1 + Amount2)/2*(20/100). The solution knowledge for this question can be expressed as:

AnyName : randomly chosen from a given list

Amount1 : ((random mod 9 +1) * 100000)

Increased_Amount : ((random mod 9 +1) * 10000)

Replace_Amount : (Amount1 - Increased_Amount)

Amount2 : (Amount1 + Increased_Amount)

Answer : ((Amount1 + Amount2)/2*(20/100).

These solution knowledge can then be translated into the following rule set to facilitate the determination of the Answer:

Increased_Amount = (random MOD 9 + 1)*10000

Increased_AmountFlag = ok

```
Amount1 = (random MOD 9 + 1)*100000

Amount1Flag = ok

IF Amount1Flag=ok AND Increased_AmountFlag=ok

THEN Amount2 = Amount1 + Increased_Amount

Amount2Flag = ok

IF Amount1Flag = ok AND Amount2Flag = ok

THEN Solution= (Amount1 + Amount2)/2*(20/100)

IF Reply =Solution

THEN Answer = Right

ELSE CALL Wrong-answer
```

Students enter their answer via the variable Reply. Thus, every student may get a different set of values for this problem through AnyName, Amount1 and Increased_Amount, and may have a different depreciation value as answer. However different the answers may be, the rule set will always ensure the correct answers to be found in the final step as well as the intermediate steps. When a wrong answer is entered, the system, through the CALL statement, may provide a guidance from the database like "You need to calculate the new plant cost for 1996 first before you can apply the straight line method which is given in page 155 of the text book",

Question 2 :

{Company1} Ltd and {Company2} Ltd have a contract in which they will jointly develop three hotels as part of a holiday resort on the Golden Coast via, upon completion, a joint venture vehicle {Company3}. {Company1} contributed 66% to the assets of {Company3}. {Company2} contributed 33% to the assets of {Company3}. The contract provides that {Company1} will take title over two hotels and {Company2} over one hotel. {Company1} Ltd and {Company2} Ltd are unrelated entities. Should {Company1} consolidate {Company3}? (yes/no/do not know)

The following rules are provided for the system to match the Reply from the user and continue into next phase of the question :

```
If Reply="yes"

Then Call Right-answer-1

If Reply="no"

Then Call Right-answer-2

If Reply="do not know"

Then Call Wrong-answer
```

In this question, depending on the different regulations used, both "yes" and "no" can be correct answers. The system can then activate the next state for another related question, which is a continuation of the previous question. For instant:

State Right-answer-1:

You are possibly correct!

For {Company1} to consolidate {Company3} it must first be shown that a joint venture is an entity under AASB 1024. Why do you think {Company3} is an entity?

To this question, any answer that uses the term "legal", "administrative", "organizational structure", or "other party" is a correct answer. The following rules are provided for the system to match the Reply from the user :

```
default = true

If "legal" in Reply

Then Call Right-answer

default= false

If "administrative" in Reply

Then Call Right-answer

default= false

If "organisational structure" in Reply

Then Call Right-answer

default= false

If "other party" in Reply

Then Call Right-answer

default= false

If default=true

Then Call Wrong-answer
```

From this goal, the system can either activate the routine that displays the message to congratulate the student on their correct answers or to provide explanation for student who typed in the wrong reply.

Conclusions

This system has been in operation for two large subjects for more than 2 semesters, the responses from students as well as lecturers are very encouraging. The most favorable feedback from lecturers is the saving of time on marking examinations. The major obstacle in using this system is the fact that first time users need to spend time, much more than the amount needed for the traditional approach, in developing questions to target specific topics and for examinations. In terms of the enhancement of the teaching-learning function, the general conclusion is that it depends heavily on the ways questions are designed and structured, and it needs good lecturers and investment of time to continue to build up a development with depth. At present, lecturers tend to use the system to develop multiple-choice questions, which are the easiest and the least time consuming. Hence, in order to better utilize this system, incentives of some form from the department may be necessary. One drawback that was pointed out in the survey is the fact that, the present system does not allow students to go back to rework on the previous questions, and this may cause some anxieties. This problem can be solved by saving the initial values of each question for later use.

(References available upon request from first author)