A MULTI-LEVEL DYNAMIC INSTRUCTIONAL PLANNER FOR AN INTELLIGENT PHYSIOLOGY TUTORING SYSTEM

CHONG W. WOO

GENERATION OF COMPLEX TUTORIAL DIALOGUES
MARTHA W. EVENS, PRINCIPAL INVESTIGATOR
COMPUTER SCIENCE DEPARTMENT
ILLINOIS INSTITUTE OF TECHNOLOGY
CHICAGO, ILLINOIS 60616
312-567-5153, csevens@minna.iit.edu

This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014-89-J-1952, Grant Authority Identification Number NR4422554, to Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

Approved for Public Release; Distribution Unlimited
# A Multi-Level Dynamic Instructional Planner for an Intelligent Physiology Tutoring System

Chong W. Woo

Dr. Martha Evens  
Computer Science Department  
Illinois Institute of Technology  
10 West 31st Street  
Chicago, IL 60616

Dr. Susan Chipman  
Cognitive Science Program  
Office of Naval Research  
800 North Quincy Street  
Arlington, VA 22217-5000

Approved for public release; distribution unlimited
A MULTI LEVEL DYNAMIC INSTRUCTIONAL PLANNER FOR AN INTELLIGENT PHYSIOLOGY TUTORING SYSTEM

Chong Woo
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois 60616
Phone: (312) 567-5153
E-Mail: csevens@minna.iit.edu

1. INTRODUCTION

An instructional planner in an intelligent tutoring system (ITS) is responsible for deciding what to do next at each step during the tutoring session. The planner has to decide what subject matter to focus on, how to present it to the student and when to interrupt the student's problem-solving activity [Dede, 1986; Kearsley, 1987]. This pedagogical decision making is very complex and there is no one correct choice due to the dynamic changes in the student's learning state. Hence, the decision must be based on many different knowledge sources, such as knowledge about the domain, knowledge about the student, and pedagogical knowledge about tutoring.

Recent approaches to designing tutoring systems view the decision making process as a planning problem [Peachey and McCalla, 1986; Macmillan et al., 1988; Brecht et al., 1989; Murray, 1990]. Adaptive planning techniques in the tutoring domain enable the generation of customized plans for individualized instruction. Among the recent research systems, MENO-TUTOR [Woolf, 1984] represents an important attempt at planning the discourse strategies observed in human tutors, but it lacks global lesson goals [Murray, 1988]. IDE-INTERPRETER [Russell, 1988] is another attempt at planning the lesson goals at various levels of abstraction, but this system lacks power at the local diagnostic level. Thus, there is a need to build an instructional planner that combines globally coherent lesson goals with flexible local discourse plans.

In this research, I am building a planner that integrates opportunistic control with a sophisticated instructional planning methods; combining capabilities of lesson planning with discourse planning. This planner is a dynamic instructional planner that supports customized, globally coherent instruction, carries out a mixed initiative strategy. It monitors current plans in progress, repairs those plans, or replans as needed. This has required the invention of multi-level instructional planning.

The goal of this research is to develop an ITS, CIRCSIM-TUTOR, that assists first year medical students to learn the behavior of the cardiovascular reflex system that stabilizes blood pressure. Since the students have already attended lectures about the domain, CIRCSIM-TUTOR assumes prerequisite knowledge and assists them to correct their misconceptions in a problem solving environment. This system is being developed as a joint project of Rush Medical College and Illinois Institute of Technology.

1.1 Evolution of Computer-Based Instruction at Rush

Computer Aided Instruction (CAI) in the cardiovascular domain at Rush Medical College has evolved from HEARTSIM [Rovick and Brenner, 1983], to CIRCSIM [Rovick and Michael, 1986], to the CIRCSIM-TUTOR prototype [Kim et al., 1989] and finally to CIRCSIM-TUTOR over the last ten years.

HEARTSIM was a Plato program and CIRCSIM is a stand-alone Basic program. The CIRCSIM-TUTOR prototype is a Prolog prototype of our ITS designed and implemented by Kim [1989]. Its design is based on major ITS architecture. However, the prototype system still does not possess all of the capabilities needed for an ITS. It lacks natural language capabilities, it does not analyze the student's misconceptions, and the instructional planner is very primitive;
a discourse planner could not be implemented since complete discourse strategies for all the primitive actions had not been developed, planning knowledge is not explicitly represented as a separate module, and there was no replanning capability so that the system could not respond to student initiatives. CIRCSIM-TUTOR uses the same architecture as Kim’s prototype but includes complete student modelling, instructional planning, and natural language understanding and generation facilities.

1.2 Organization

Section 2 describes the environment in which the system runs. The subject area of CIRCSIM-TUTOR is cardiovascular physiology and the system assists students to understand the behavior of the complex negative feedback system. Section 3 begins with a brief introduction to ITS: the general structure and the issues involved in each module of the ITS. Then each component of CIRCSIM-TUTOR will be briefly introduced. Section 4 presents design issues for building the planner: levels of planning and tutoring strategies. A short tutoring excerpt is displayed, from a transcript of human tutor and student interaction. And then a short scenario shows how the system works. It concludes with a discussion of the overall organization of the planner: lesson planning, discourse planning, and plan monitoring. Section 5 explains the generation of the content of lesson plans in detail. It first discusses the main features of the planner: goal generation and plan generation. And then it describes its own lesson planning rules: goal generation rules and plan expansion rules. Section 6 discusses the discourse planner. The structure of the planner is a two level discourse management network, which consists of a set of states that represent tutorial actions. The control mechanism is separated into default and meta-rule transitions. The paper concludes in section 7 with a discussion of the significance of the planner, describes some of its limitations, and gives suggestions for future research.

2. THE BACKGROUND

Qualitative reasoning or simulation [deKleer and Brown, 1984; Forbus, 1984; Kuipers, 1984] is an approach to problem solving that reasons about the causal relationships that structure our world. Anderson [1988] argues that qualitative reasoning is the most demanding approach and essential to produce a high performance tutoring system. He states that qualitative modelling can maximize the pedagogical effectiveness since it is human-like reasoning, although the implementation effort is much larger than that required for the traditional black box models or glass box models. CIRCSIM-TUTOR is an approach to qualitative simulation in cardiovascular physiology [Michael et al., 1990].

2.1 Subject Area

The cardiovascular system consists of many mutually interacting components, and the student must understand the cause and effect relationships for each individual component of the system. Figure 1 shows a causal model of CIRCSIM-TUTOR, called the "Concept Map," designed by Michael and Rovick [Kim et al., 1989]. Each box in the map represents a physiological variable, such as SV for Stroke Volume and RAP for Right Atrial Pressure. An arrow with a "+" or "−" sign between two boxes tells the direction of the causal effects and whether the causal relationship between the connected variables is direct or inverse. For example, a qualitative change in one component of the system, a decrease in RAP, directly causes a decrease in SV. This qualitative change propagates to other adjacent components of the system according to the propagation rule.

There are three stages in the human body's response to a perturbation in the system that controls blood pressure. The first stage is the Direct Response (DR) in which a perturbation in the system will physically affect
many other parameters. The second stage is the Reflex Response (RR), in which other parameters are affected by the negative feedback mechanism to stabilize the blood pressure. The final stage is the Steady State (SS), which is achieved as a balance between the changes directly caused by the initial perturbation and the further changes induced by negative feedback.

Figure 1. The Concept Map

2.2 Organization

CIRCSIM-TUTOR begins with a brief introductory message and then asks the student to choose any procedure from the curriculum list. The curriculum (Figure 2) is stored as a set of four different experimental procedures designed by our expert human tutors (JAM and AAR).

List of Available Procedures

1. Hemorrhage: Remove 1.0 Liter of Blood.
2. Decrease Cardiac Contractility (CC) to 50% of Normal.
3. Increase Venous Resistance (RV) to 200% of Normal.
4. Increase Intrathoracic Pressure (PIT) to 2 mmHg.
5. Quit.

Figure 2. List of Available Procedures
Each procedure begins by describing a perturbation of the cardiovascular system, and asking the student to predict how the system variables will respond to the perturbation by making qualitative entries in the Prediction Table (see Figure 3); using a "+" sign to represent an increase, a "-" for a decrease, and "0" to indicate no change. The first column of the table is used to predict the Direct Response (DR) of each variable to the perturbation, the second is used for the Reflex Responses (RR), and the third for the Steady State (SS).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DR</th>
<th>RR</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardiac Contractility</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Right Atrial Pressure</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Stroke Volume</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Heart Rate</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cardiac Output</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Total Peripheral Resistance</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Mean Arterial Pressure</td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 3. The Prediction Table

When the student finishes predicting all four parameters in one column of the table, for example the DR stage, the student's answers are compared with the correct answers. If the student has made any errors, a natural language tutoring session will begin, based on the result of this evaluation in order to correct the student's misconceptions.

2.3 System Constraints

There are some system variables that need to be described; the procedure variable is the variable changed by the perturbation; the primary variable is the first variable in the Prediction Table affected by the procedure variable, (in some cases the procedure variable is the primary variable); the neural variables are the variables directly under nervous system control. The rest of the variables we call physical variables. The students are not allowed to predict the variables in any arbitrary order, since there are some constraints that they must follow. For example, the constraints for DR are fairly complex:

- **Constraint DR1:** The student must predict the primary variable first, and the value must be correct.
- **Constraint DR2:** The student must predict the physical variables in the correct causal sequence.
- **Constraint DR3:** The student may predict the neural variables at any time and in any order.
The student receives a canned error message, when either of the first two constraints is violated, and is told what to do next. The purpose of forcing the student into the correct sequence is to make sure the causal behavior of the system is followed correctly. Neural variables can be entered at any time since neural variables do not change during the DR period except when one is a primary variable. The constraints for the RR stage are designed to teach the students about the effect of the baroreceptor reflex:

- **Constraint RR1:** The student must predict either the neural variables or MAP first.
- **Constraint RR2:** The student must finish predicting all the neural variables before predicting other physical variables.
- **Constraint RR3:** The student must predict the physical variables in the correct causal sequence.

Finally, when predicting the SS stage, the student is allowed to enter predictions in any arbitrary order since there are no specific constraints for this stage.

### 2.4 Multiple Simultaneous Inputs

In a mixed-initiative type of ITS, tutor begins by posing a question and the student either responds to the question or takes the initiative. Sometimes this style of tutoring leaves students confused and frustrated if they do not have enough background in the domain knowledge. Rather than blindly walking through the domain, it would be much more effective if the tutor provides a simulated problem situation in the domain for the student before the actual interactive tutoring begins.

CIRCSIM-TUTOR begins with a Prediction Table, in which the student is asked to make qualitative predictions about the behavior of the system given a particular perturbation. After the student finishes all the predictions, the tutor analyzes the student's answers and shows what errors were made if any. Based on a careful analysis of these errors, the tutor can generate a global lesson plan, and interactive tutoring begins by using a mixed-initiative Socratic strategy in natural language. Thus, the Prediction Table provides a qualitative simulation environment for the student by requiring multiple simultaneous inputs (multiple responses to different aspects of a problem provided by the student in a single uninterrupted turn) before interactive tutoring begins.

There are several benefits of adapting this kind of design strategy. First, the tutor receives enough initial knowledge about the student so that it can narrow the focus for tutoring. It can also detect some common student misconceptions [Michael et al., 1991] or bugs. Second, the students can see a simple mental model of the entire domain at the start, which prevents the students from getting too far off the track [Reiser, 1989]. Elsom-Cook [1988] argues that using multiple pedagogic strategies can provide a very powerful learning environment. CIRCSIM-TUTOR begins with a coach-like environment during the Prediction Table entry, and then moves to Socratic tutoring for the interactive tutoring session. This flexibility in adapting to the student's needs at different stages provides another benefit.

### 3. ORGANIZATION OF CIRCSIM-TUTOR

Most of ICAI systems have been separated into four major components [Carr and Goldstein, 1977; Sleeman and Brown, 1982; Barr and Feigenbaum, 1982]: the domain knowledge base, a collection of instructional strategies and an algorithm for applying them, a student modeler, and an interface. Since a major goal of CIRCSIM-TUTOR is to carry on a natural language dialogue, we have divided the interface module into three pieces, an input understander, a
text generator, and a screen manager. As a result, CIRCSIM-TUTOR has seven submodules: a domain knowledge base, a problem solver, a student modeler, an instructional planner, an input understander, a text generator, and a screen manager. Figure 5 shows the overall architecture of CIRCSIM-TUTOR.

3.1 Domain Expertise

Domain Knowledge Base. Anderson [1988] describes three different categories of knowledge encoding: the black box model, the glass box model, and the cognitive model. The cognitive model is the approach that CIRCSIM-TUTOR is attempting to implement. The domain knowledge is decomposed into meaningful, human-like components and a causal reasoning mechanism is applied to it, so that the system can teach the student to solve problems in a human-like manner. For a detailed discussion of this problem see Wielinga and Breuker [1990].

Domain knowledge can be divided into three different types of knowledge to be tutored: declarative knowledge, procedural knowledge, and knowledge of tutoring heuristics. Declarative knowledge includes domain concepts and causal relationships between them. Procedural knowledge involves the rules for using the concepts in solving problems. For example, in CIRCSIM-TUTOR, a rule that figures out the actual determinant of SV is if the primary variable is RAP, then RAP is the actual determinant of SV. Knowledge of tutoring heuristics must be extracted from the experience of domain experts; it involves ways of teaching the student about the particularly difficult points in the domain.

We have built a small domain knowledge base encoded as a network of frames (see Figure 4). Each frame represents domain concepts and how they relate to each other causally. There are three conceptual levels in the domain knowledge; level 0 consists of the definitions and static facts, level 1 consists of the cause-effect relationships between the parameters of the cardiovascular system, and level 2 contains a deeper knowledge of underlying physiology. The level 2 knowledge is used when the tutor needs to give a hint to the student. Currently, the level 2 knowledge is under refinement and development. Hence, in the present program the domain knowledge base is constructed as a set of components that is used for both problem solving and causal explanation. This is the most important and the basic knowledge that constitutes the domain expertise.

```
(frame SV
  (frame-type variable
  var-type physically-affected
  frame-name SV
  class instance
  instance-of variable
  name Stroke Volume
  definition volume of blood ejected each heart beat
  part-of heart
  anatomy ventricle
  causal-relation-in causal-RAP-SV causal-CC-SV
  causal-relation-out causal-SV-CO))
```

Figure 4. A Frame from the Domain Knowledge Base

Problem Solver. The intelligence of an ITS comes from its ability to solve the problems [Clancey, 1987]. The problem solver solves the problems
presented to the student or asked by the student. If the problem solver solves
the problems but can not explain how it solves them, it may just as well
retrieve stored answers. The ability to solve the problem, using the expert's
problem solving behavior, can be used to identify the student's
misconceptions, to give an explanation, and to provide a basis for tutoring
strategies.

Problem solving in CIRCSIM-TUTOR is carried out by two problem solvers:
the main problem solver and the subproblem solver. The main problem solver
solves the problem, generates correct answers, and produces the same problem
solving path as an expert in the domain. This solution path can be used to
monitor the student's problem solving behavior while the student is making
entries in the predictions table. The subproblem solver solves current
problems generated by the planner, such as determinant of \( X \), relationship
between \( X \) and \( Y \), and also problems coming from the student questions. The
other modules of the system may consult these problem solvers to get any
information they need.

3.2 Student Modeler

The student modeler is responsible for representing the student's
understanding of the subject by building a student model [VanLehn, 1988]. The
student model is a data structure that represents the student's current state
of knowledge; what the student knows, what the student does not know, and what
misconceptions he or she may have. Based on this information, the tutor can
give individualized instruction to the student. There are two major approaches
for student modeling. One approach, the overlay model [Carr and Goldstein,
1977], is designed to represent the student's knowledge state as a subset of
an expert's knowledge state. Another approach, the buggy model [Brown and
Burton 1978], represents the student's misconceptions not as subsets of the
expert's knowledge, but as variants of the expert's knowledge. In CIRCSIM-
TUTOR, the student modeler integrates overlay and buggy strategies into one
[Shim et al., 1991].

The student modeler begins analyzing the student's entries in the
Prediction Table. Based on this analysis, the planner generates a lesson plan.
During the tutoring session, the planner sends the student's answer to the
modeler and the modeler analyzes it and returns the result. Based on this
information, the planner can decide the next instruction. Currently only the
overlay information is used for choosing the next tutoring strategy.

3.3 Input Understander

The input understander is responsible for understanding the student's
natural language input. It handles not only well-formed but also ill-formed
student inputs [Lee et al., 1990; Lee, 1990]. The student input may be either
an answer to the tutor's question, or a question from the student. If the
student's answer is *The actual determinant of SV is RAP*, then the planner will
pass the sentence to the understander along with the current lesson topic in
logical form, (actual-determinant SV). Then the input understander parses the
sentence, checks its coherence with the current topic, and returns the logic
form, (answer (actual-determinant SV (RAP))). Then the planner extracts the
student answer, RAP, and passes it to the student modeler to diagnose the
student answer.

The input understander must also understand student initiatives; whether
the student is asking for an explanation, or referring to the previous remarks
of the tutor, or wants to stop the session. For example, if the student
initiative is *I don't understand about SV*, then the input understander returns
the logical form (question (explain SV)). This process needs to be studied in
detail and we are currently investigating the student initiatives by analyzing
transcripts.
3.4 Text Generator

The text generator is responsible for turning the tutor's output into a natural language sentence. It receives necessary information as a logical form from the planner and generates a natural language sentence or sequence of sentences [Zhang and Evens, 1990]. This information includes the current topic and text styles: question, hint, answer, etc. For example, the text generator is given a logic form from the planner, \((\text{question} \, \text{affected-by} \, \text{SV} ?)\), then it produces the English sentence, "What are the determinants of SV?" The text generator can handle this kind of simple question, explanation, or acknowledgement. The current version of the text generator only receives the necessary information from the planner, not from all the other modules, so that its behavior is somewhat passive.

3.5 Screen Manager

The screen manager takes care of the interaction between the student and the system. First, the screen manager displays system messages through the introductory windows. Then it displays the list of procedures that the student can select. When the student selects the problem, it displays the prediction table with instructions about how to use the mouse and how to make entries into the table. Then it receives qualitative answers, (+, -, 0), from the prediction table one by one from the clicking of the mouse and passes them to the planner. It also handles two other windows, the student window and the tutor window. From the student window, it receives the student's natural language input in English sentences. In the tutor window, it displays natural language sentences created and passed to it from the text generator.

3.6 Instructional Planner

The instructional planner is responsible for determining what to do next at each point during a tutoring session. It interacts with the input underadarnder, the text generator, the student modeler, and the screen manager, in order to carry out tutorial activities. Although the design of the planner may vary depending on the purpose of the ITS, several researchers have recently proposed combining opportunistic control with a plan-based approach [Derry et al., 1988; Murray, 1990; Macmillan and Sleeman. 1987]. For instance, Murray [1990] suggests that the way to provide opportunistic control with global lesson plans is to implement a dynamic instructional planner. For CIRCSIM-TUTOR, the planner needs to generate the global lesson plan and take care of the discourse control as well [Woo et al., 1991a, 1991b].

4. Planning Instruction

The instructional planner is the central component of the ITS; it is responsible for selecting or generating instructional goals, deciding how to teach the selected goals, monitoring and critiquing the student's behavior, and determining what to do next at each point during a tutoring session. That is, the planner makes two different types of important decisions during the tutoring session, decisions about the content of the lesson and decisions about its presentation strategy. Although the early ITSs largely focused on the delivery strategy of the planner, some recent planning research shows the integration of both aspects in building the planner [Macmillan et al., 1988; Derry et al., 1988; Murray, 1990].

The planning component of CIRCSIM-TUTOR must carry out both functions, since it needs to provide a global lesson plan, and it needs to carry on a natural language exchange with the student. This section discusses general design issues of the planner with the goal of providing the most effective instruction possible to the student, a sample dialogue extracted from the transcript of an actual human tutor-student interaction and a scenario implementing that dialogue, and a description of the overall organization of the planner.
4.1 Design Issues

Capabilities of the Planner. Most machine planning systems, like STRIPS [Fikes and Nilsson, 1971], HACKER [Sussman, 1975], and NOAH [Sacerdoti, 1977], deal with the observable physical world, whereas instructional planning systems deal with unpredictable dynamic changes in the student’s knowledge. The student’s current learning status can never be observed directly. It can only be guessed; the results of this guesswork are stored in the form of the student model [VanLehn, 1988]. Thus, the planner must possess unique capabilities for handling unpredictable situations as an expert human tutor does.

The planner must plan at different levels of the hierarchy; a hierarchical planning technique can reduce the complexity of the planning process. The plan is first developed at a higher level and the details are developed later; this technique prevents development of unnecessary plans in
The planner must plan at a global level; when the planner generates the next instruction, it must consider the past plan and the student's responses to provide continuity of instruction. The planner must replan when the current plan fails or a request is made by the student. The planner must be able to monitor the plan to identify the need for replanning. The planner of CIRCSIM-TUTOR provides all these capabilities. Levels of Planning. Research by Leinhardt and Greeno (1986, cited in Derry et al., 1988) has shown that experienced teachers employ levels of planning in accomplishing their goals; planning instructional goals occurs at the most global level, planning actions and decision making occur at a less global level. Inspired by this research, Derry et al. (1988) designed their TAPS system with three levels of instructional activity: curriculum planning (the agenda), lesson planning (instructional actions), and on-line tutorial intervention. Murray (1988) also distinguished three levels of instructional planning: curriculum planning (planning a sequence of lessons), lesson planning (determining the subject matter in a single lesson), and discourse planning (planning communicative actions between the tutor and the student). He argues that at least two levels of planning, lesson planning and discourse planning, must exist in an ITS to deliver more effective and flexible instruction.

CIRCSIM-TUTOR is capable of both lesson planning and discourse planning. It can be set up so that the student can select a problem from a list of four experimental procedures or it can do complex curriculum planning. The number and types of procedures will be extended further in future versions of the system.

4.2 Scenario

A Sample Tutoring Session. We have recorded a number of tutoring sessions with our experts, Joel Michael and Allen Rovick, who are Professors of Physiology at Rush Medical College, and some of their first year medical students. After careful studies of the recorded transcripts, we extracted some possible tutorial strategies and tactics that provided us with the framework for building the instructional planner and the overall system. It is assumed that students have already learned much of the domain knowledge, hence the system will mainly assist the students to correct their misconceptions and to solve problems. Our current system can handle dialogues like the following.

Example 1:

Tutor> What are the determinants of SV?
Student> SV is determined by RAP and CO.
Tutor> RAP is correct, but CO is not a determinant of SV. Remember, SV is the amount of blood pumped per beat. What is the other determinant of SV?

One important point about the above tutor-student interaction is the content of the questions posed by the tutor. For example, on the first line of the excerpt, the tutor is asking the student about the determinants of stroke volume. Asking a question about determinants is the first part of the plan that the tutor is using to teach the student about the causal relationships between two variables, RAP and SV. Thus the content of the question has to be generated by the lesson planner before the tutoring begins. Another important aspect is how to present the selected topic. From the above short excerpt, we can see four different kinds of delivery modes: a direct question (line 1), positive and negative acknowledgements (line 3), hints (line 4), and follow up questions after hints (line 5). Thus, the planner (discourse planner) needs to plan how to present the selected content to the student effectively.

Example 2:

Tutor> By what mechanism is TPR controlled?
Student> Nervous System.
Tutor> Correct, TPR is controlled by the nervous system.
Then what is the correct prediction of TPR?

Student> No change.

Example 2 is another tutoring situation that focusses on one of the neurally controlled variables, TPR. The tutor first asks the student about its control mechanism in line 1. This control mechanism is the first strategy to teach the student about the neurally controlled variables. Since the student answered correctly, the tutor gives a positive acknowledgement and then uses its second strategy, asking for a prediction, in line 4. We have extracted this kind of tutoring strategy from the transcripts and designed explicit lesson planning rules.

From the above examples of tutor-student interaction, we can distinguish between the subject matter and its presentation formats. Ohlsson [1986, p. 217] argues that an effective ITS should be able to generate different presentations of each piece of subject matter in order to provide adaptive instruction to the student. The content of the questions posed by the tutor and its delivery modes lead to the development of two different kinds of instructional planning, lesson planning and discourse planning, because the subject to be taught has to be generated adaptively, and also its presentation form can vary according to the situation.

Implementation of the Scenario. Assume that the current lesson goal is to tutor the causal relationships between two parameters, RAP and SV. This goal gets refined into a set of hierarchical subgoals by using strategic and tactical rules. The subgoals generated at the tactical level, such as determinants, actual determinant, relation, and value, are kept in a stack, which is used by the discourse planner to pick the next topic.

The following scenario describes what each component of the system does, what kind of information it needs, and what is the result after each step. The steps are numbered to show the execution sequence. This tutorial interaction begins after the lesson planning is done. So that the discourse planner begins with the first topic in the stack, the determinants, and when that topic is completed, continues with the next topic, the actual determinant, and so on.

1. Planner: Picks the current topic from the stack, selects the discourse tactic, and passes it to the text generator as an internal logical form.

   current topic: (determinant SV),
   discourse tactic: question.
   call Text Generator: (question (determinant SV))

2. Text Generator: Generates a sentence, "What are the determinants of SV?"

3. Screen Manager: Displays the sentence in the window.

4. STUDENT: "SV is determined by RAP and CO."

5. Planner: Passes the student's input with the current lesson topic to the input understander.

   (question (determinant SV)
   (SV is determined by RAP and CO))

6. Input Understander: Parses the student's answer, checks its coherence with the dialog history, and returns the answer to the planner in logic form.

   call planner: (answer ((determinant SV) (RAP CO)))
7. Planner: Passes the current topic and student answer to the student modeler in logic form.

   current topic: (determinant SV),
   student answer: (RAP, CO),
   call Student Modeler: ((determinant SV) (RAP, CO))

8. Student Modeler: Calls the problem solver, gets the correct answer: (RAP, CC), compares the correct answer with the student answer, and updates the student model.

   In step 1, the discourse planner picks the topic, determinant, from the subgoal stack, selects the discourse tactic, question, binds these two together with the current variable, SV, into a logical form, (question (determinant SV)), which is passed to the text generator to generate a natural language sentence. After receiving the logical form from the planner, the text generator generates a sentence like the one in step 2. In step 3, the screen manager displays the sentence on the student window, and the student responds with the answer in step 4. So the current dialogue is:

   Tutor> What are the determinants of SV?
   Student> RAP and CO.

   In step 5, the planner passes the student's input along with the current topic. The input understander has to recognize the student's answer; parse the answer, check its coherence with the dialogue history, and return the answer to the planner in its logical form. Then the planner sends the current topic with the student's answer to the student modeler in step 7. Finally, the student modeler analyzes the student's answer, and records the result in the student model. The next step will start with the planner checking the student model, and then deciding what to do. Since one of the student's answers is wrong, the planner consults its tutoring rules and decides to give some acknowledgement first:

   Tutor> RAP is correct, but CO is not a determinant of SV.

   At this point the tutor has two choices, either give a hint or just give an answer and continue with the next topic. Since this is the first trial, the tutor decides to "give a hint" and then ask a question to complete the previous answer. So a possible response would be:

   Tutor> Remember. SV is the amount of blood pumped per beat. What is the other determinant of SV?

   A different tutoring rule will be applied if the student again makes an error after receiving a hint; the student will be given a direct answer for the second question. Our current tutoring rules vary according to the topic and the student's responses (i.e., the tutor gives different responses in different situations). The question may be about neural variables or causal relationships; in each case the tutoring rules are different. Also we have different rules for each stage, DR, RR and SS.

4.3 Organization of the Instructional Planner

   The instructional planner of CIRCSIM-TUTOR consists of two parts (see Figure 6); the lesson planner and the discourse planner. The plan controller monitors the execution of the current plan. The planner can be thought of as a small expert system, which consists of two main parts: a knowledge base and an
inference engine [Harmon, 1987]. Thus, the lesson planner consists of three sets of lesson planning rules, and an inference mechanism. The discourse planner consists of four sets of discourse planning rules and an inference mechanism. This section introduces the organization and the main features of the instructional planner briefly.

INSTRUCTIONAL PLANNER

Lesson Planning. Lesson planning determines the content and sequence of the subject matter to be taught in a single lesson [Murray, 1988; Brecht et al., 1989; Russell, 1988]. The lesson planning in CIRCSIM-TUTOR consists of two phases: goal generation and plan generation. The generation of the lesson
goals is guided by a set of explicit domain-dependent heuristics (goal generation rules), and the lesson plans are determined by applying two sets of rules, rules for selecting strategies and rules for selecting tactics. As a result the lesson planner does hierarchical lesson planning with its three sets of rules; at the topmost level it generates lesson goals, and then it expands one of the goals into a set of subgoals (a plan) at the next level. The generated goals will be saved in the goal stack and the subgoals in the subgoal stack. The lesson planner must update the goals dynamically as the student model changes.

Discourse Planning. Discourse planning is a mechanism for planning communicative actions between the tutor and the student within a lesson [Woolf, 1984; Winkels et al., 1988]. CIRCSIM-TUTOR communicates with the student in natural language. Thus, the discourse planner must interact with the student modeler, the screen manager, the input understander, and the text generator using a flexible control mechanism. This control mechanism resides in its discourse network.

The network consists of two levels; the top level of the network specifies pedagogic decisions and the lower level consists of a set of discourse tactical states, the execution of which causes text generation, student model updates, and moves to the other states. It represents the discourse planning rules and the control mechanism in explicit form. The rules include all the necessary information to carry out the discourse with the student, and the control mechanism is also specified within the rules; two sets of default rules manage the fixed control flow, and two sets of meta rules handle dynamic control flow.

Plan Monitoring. AI research on planning emphasizes that execution of a plan requires some monitoring [Charniak and McDermott, 1986]. In the recent robot planning systems [Wilkins, 1988; Swartout, 1988], the plan monitoring is done by inserting monitoring steps in the plan, which behaves like a student model in instructional planning. In an ITS, since the student's learning status is unpredictable, the planner also needs to monitor the execution of the plan and revise the plan if necessary. As a result, plan monitoring should occur whenever there is a change in the student model. Plan revision may occur when the current plan is completed or when the student takes the initiative.

For the current version of CIRCSIM-TUTOR, the planner monitors the student problem solving in two different places. First, when the student enters predictions in the prediction table, the planner monitors the student's entries in the table and interrupts with some warning messages if the student violates the system constraints. The messages are designed by the experts, to help the students in their problem solving. The system gives different messages depending on the procedure, the variables, and the stages. Second, the planner monitors the student answer at each step during the tutoring session, by referring to the student model, and then decides what to say next; give a hint, give an answer, or continue with the next topic, etc. When the student takes control by asking a question during the tutoring session, the planner suspends the current plan, carries out the student's request and then resumes the suspended plan.

5. THE LESSON PLANNER

The lesson planner decides on the contents of a lesson, based on the student's current knowledge about the domain. The planner has to generate lesson goals, sequence the goals, and select the appropriate planning strategies to create a plan for the current lesson goal. Figure 7 shows the architecture of the lesson planner including the necessary planning steps, student model, and lesson planning rules. The result of the lesson planning is a set of subgoals (a plan), each of which will be the topic for a dialogue with the student.
The lesson planning mechanism is an essential component of the instructional planner, since the system must generate globally coherent and consistent instruction for the student (Macmillan et al., 1988; Murray, 1990), in such a way that the topics are logically connected with each other, and sequenced and presented in a manner sensitive to the tutorial goals and the student's needs. This section describes the lesson planner: lesson planning rules, an architecture, and its two main mechanisms (i.e., goal generation and plan generation).

**LESSON PLANNER**

![Diagram](image)

**5.1 Lesson Planning Rules**

The contents of the tutoring strategies are extracted from the transcripts of the human tutor and student interaction, and we need to encode them explicitly in the program as rules. I designed this part as a production system, which consists of a rule interpreter and a set of rules. This is the most common approach used in expert systems [Hasemer and Dominque, 1989]. This subsection, describes the design of the rules and the implementation of the rule interpreter to parse the rules.

The Rule Interpreter. The rule interpreter consists of three main parts: its main loop, its working memory, and its pattern matcher. The working memory is crucial to the operation of the rule interpreter, because the working memory holds an initial representation of the problem that the system is trying to solve. Each time around the loop, the contents of the working memory will be compared to the antecedent of the rules, and then will fire only one rule if it matches. If an antecedent matches with the working memory, the consequent will be executed, and the content of the working memory will be changed for the next inference. The matching cycle will continue until no rules match. At this point the interpreter halts, and the content of the working memory is the desired result for the given problem.

The interpreter is built using LISP macro functions, which understand and interpret the rules for the system. The rule format consists of three parts: the name part of the rule, the antecedent part, and the consequent part. For example, \( \text{Rule name: (antecedent)} \Rightarrow \text{(consequent)} \). This approach makes the system efficient in representing the rules explicitly.

How to Encode the Lesson Planning Rules. The lesson planner uses three sets of lesson planning rules (goal generation rules, strategy rules, and tactical rules). The general form in which the rules are written is if \( X \) then \( Y \). Here \( X \) is the antecedent or left-hand side of the rule and \( Y \) the consequent part or right-hand side of the rule. Both the antecedent and the consequent may contain one or more terms.

For example, assume that the student made an error in predicting the variable TPR. One of the goal generation rules applies; if the student does not know TPR, then build the lesson goal, tutor TPR about the neural control.
This rule can be expressed as \((G_{Rule1}: ((\text{do-not-know TPR}) \Rightarrow (\text{neural-control TPR}))\)). If the current lesson goal is teach the causal relationship between RAP and SV, and the student does not know the direction, then this rule can be written as \((S_{Rule1}: ((\text{causal-relation})(\text{do-not-know direction})) \Rightarrow (\text{tutor-causality})))\). This is the strategy rule for dealing with non-neural variables. If the strategy rule is tutor-causality, then the corresponding tactical rule is to teach determinants, actual-determinant, relation, and value. This rule can be written as \((T_{Rule1}: ((\text{tutor-causality}) \Rightarrow (\text{determinants}) (\text{actual-determinant}) (\text{relation}) (\text{value}))\)).

Currently, there are about 50 goal generation rules, 20 strategy rules, and 20 tactical rules that handle DR, RR and SS phases, and for procedures 4, 6 and 7. The rules may need to be extended to handle the other procedures.

5.2 Lesson Planning

Instructional planning centers around instructional goals. The lesson planning generates the lesson goals, the knowledge that the system intends the student to acquire through the tutoring session. This subsection describes how to generate the lesson goals, and how to develop a lesson plan for each of the goals. The two main mechanisms of the lesson planning process, goal generation and plan generation, are explained below.

**Goal Generation.** CIRCSIM-TUTOR generates instructional goals based on the student's knowledge demonstrated as entries in the Prediction Table. The generation of the goals is guided by a set of explicit goal generation rules designed by our experts (Joel Michael and Allen Rovick), which ensures that the most serious misconception is selected and tutored first.

### Goal Generation Rules

1. IF Current Primary Variable is CC and Student Answer is not NOCHANGE for TPR
   Then Build Lesson Goal (NEURAL-CONTROL (TPR))

2. IF Current Primary Variable is RAP and Student does not know the CAUSAL-RELATIONSHIP between RAP and SV
   Then Build Lesson Goal (CAUSAL-RELATION (RAP, SV))

3. IF Current Primary Variable is RAP and Student does not know the CAUSAL-RELATIONSHIP between SV and CO
   Then Build Lesson Goal (CAUSAL-RELATION (SV, CO))

**Figure 8. Goal Generation Rules**

For example, suppose the student made wrong predictions in the table for the variables, TPR and SV. The student modeler has determined, from its analysis, that the student is confused about the mechanism controlling TPR and
the causal relationships between RAP and SV and SV and CO. So the lesson planner retrieves the information from the student model, applies the goal generation rules (see Figure 8), and generates the lesson goals dynamically. The result is a set of lesson goals in the goal stack (see Figure 9).

<table>
<thead>
<tr>
<th>Order</th>
<th>Lesson Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>NEURAL-CONTROL (TPR)</td>
</tr>
<tr>
<td>2.</td>
<td>CAUSAL-RELATION (RAP, SV)</td>
</tr>
<tr>
<td>3.</td>
<td>CAUSAL-RELATION (SV, CO)</td>
</tr>
</tbody>
</table>

Figure 9. Generated Lesson Goals in the Goalstack

The goal generation is significant in many ways; the goals are generated dynamically and adaptively; the goals are sequenced in the order that the expert tutors this material; the goals provide a global context that remains coherent and consistent throughout the tutoring session, unless the goals are revised. New goals can also be generated, which tutor the student about a common misconception (a bug), if the student modeler detects such a misconception. The goals remain in force until they are changed by the planner dynamically.

<table>
<thead>
<tr>
<th>Strategic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. If the Goal = CAUSAL-RELATION and Student does not know and direction is incorrect Then Strategy = TUTOR-CAUSALITY</td>
</tr>
<tr>
<td>2. If the Goal = CAUSAL-RELATION and Student does not know and direction is correct Then Strategy = REMIND-RELATION</td>
</tr>
<tr>
<td>3. If the Goal = NEURAL-CONTROL and this is the first procedure Then Strategy = DEFINE-TUTOR-NEURAL</td>
</tr>
</tbody>
</table>

Figure 10. The Strategy Rules

Plan Generation. The second stage of the lesson planning is the plan generation mechanism, which creates the instructional plan by applying two sets of rules, rules for selecting tutorial strategies to achieve the selected goal and rules for selecting pedagogic tactics to execute those strategies.
Strategy rules (Figure 10) describe the tutorial approach from a domain-independent point of view. These include tutoring prerequisites before the material they underlie, reminding the student about relations between two parameters, explaining the definition before tutoring about it, and so on. Tactical rules (Figure 11) also represent a domain-independent tutorial approach; they involve asking about concepts and relations between the concepts.

<table>
<thead>
<tr>
<th>Tactical Rule</th>
</tr>
</thead>
</table>
| 1. If Strategy = TUTOR-CAUSALITY  
Then Tactic = DETERMINANTS, ACTUAL-DETERMINANT, RELATIONSHIP, VALUE |
| 2. If Strategy = TUTOR-NEURAL-CONTROL  
Then Tactic = MECHANISM, VALUE |
| 3. If Strategy = TUTOR-SS-PHYSICAL-VARIABLE  
Then Tactic = VALUE-DR, VALUE-RR, VALUE-SS |

Figure 11. The Tactical Rules

For instance, if the goal is to teach the causal relationship between the two parameters, then the fired strategy rule is to tutor the causality, and this then fires the pedagogic tactical rule: ask about: determinants, actual determinant, relationship, and correct value. The result is a hierarchical goal tree (Figure 12).

Figure 12. Generated Plan for "causal_relation(RAP,SV)"
Thus the current goal is ultimately refined into four subgoals by two-step goal transformations. In order to solve the current goal, all the subgoals must be solved. This is the well-known AI problem-reduction technique, which transforms a goal into a set of immediate subproblems by a sequence of transformations [Barr and Feigenbaum, 1982]. The four subgoals generated at the tactical level are the current plan for the goal. These are kept in a subgoal stack (Figure 13), which is used by the discourse planner to pick the next topic.

<table>
<thead>
<tr>
<th>Order</th>
<th>Subgoals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Determinants</td>
</tr>
<tr>
<td>2.</td>
<td>Actual-determinant</td>
</tr>
<tr>
<td>3.</td>
<td>Relation</td>
</tr>
<tr>
<td>4.</td>
<td>Value</td>
</tr>
</tbody>
</table>

Figure 13. The Subgoal Stack

5.3 An Example

Figure 14 shows an example of the lesson planning process for the causal-relationship between RAP and SV. From the top of the Figure, the goal generation step is described with its other information: student model, rules used, goal stack, and current goal. Then the plan generation step is described in two steps, the strategic and the tactical steps. The lesson planner waits for the discourse planner to complete the current lesson plan, and when the plan controller sends a wake-up signal, then the planner gets reactivated and continues with the next goal in the goal stack.

6. THE DISCOURSE PLANNER

The discourse planner is responsible for controlling interactions between the tutor and the student. It needs to decide how the tutor should respond to a student with a given problem [Woolf, 1984; Winkels et al., 1988]. This discourse strategy must be planned explicitly by the discourse planner, so that the system can enter into flexible and coherent interactions. In CIRCSIM-TUTOR, the discourse planner is combined with the lesson planner, so that the discourse planner receives a global lesson plan from the lesson planner. The plan controller monitors the execution of the plan and forces the discourse planner to suspend or resume the current plan when the student takes control. The planner consists of sets of discourse planning rules and a two level discourse network.

6.1 Architecture of the Discourse Planner

Flow Chart Approach. Meta knowledge is knowledge about knowledge [Davis and Buchanan, 1987]; what you know and don't know (operational meta knowledge), and how you do things (control meta knowledge). The operational meta knowledge is needed to recognize a question outside the limits of the system. It can be ignored in the discourse planner, since the input understanding receives such a question or answer and responds with I don't understand, please rephrase. The control meta knowledge determines how the system interacts with the student; it is based on our observations about how the human expert tutors the student. The integration of this knowledge into the system ensures that it appears to ask questions in a logical order.
Goal Generation

Rules Used:

Student Model: do-not-know (SV)  DR_G_Rule8

Goal Stack: Causal-relation (RAP, SV)
            Causal-relation (SV, CO)

Current Goal: Causal-relation (RAP, SV)

Plan Generation

Rules Used:

Strategy: Tutor-causality  DR_S_Rule1

Tactics: (determinants)
         (actual-determinant)
         (relation)(value)

Subgoal Stack: (determinants)
              (actual-determinant)
              (relation)(value)

Discourse Planner

executes "determinants of SV"

Plan Monitoring: Waits for the student response

Figure 14. An Example of Lesson Planning

The basic representation of the control meta-knowledge in CIRCSIM-TUTOR is the flow chart. This is a model of what the expert does and when he does it. For our system, Allen Rovick designed several flow charts (see Figure 15), each of which is used for tutoring the student in a different situation. We need different tutoring strategies for handling different variables and different phases (DR/RR/SS).
Figure 15. The Flow Chart for Tutoring Non-Neural Variables in DR

Figure 15 is the one that tutors the non-neural variables in DR. The content of the questions is determined by the lesson planner and passed on to the discourse planner, which must then decide how to express this content, determining whether to ask a question, give an answer, and so on. After the chart was created, I encoded this information as discourse planning rules. The
next step is to create a sophisticated inference mechanism that can utilize these rules.

**Discourse Network.** The network is the main knowledge structure of the discourse planner. It consists of states, links, and arcs (see Figure 16). The states represent tutorial actions, the arcs imply state transitions, and the links indicate hierarchical dependencies; a state at the tactical level represents the refinements of the level above. Three important mechanisms need to be discussed: levels of planning, representation of the tutorial states, and control structures.

![Figure 16. The Discourse Network](image)

A. Levels of Planning. The discourse planner is divided into two planning levels: pedagogical and tactical. The pedagogical level makes decisions about the style of tutoring; introduces a topic, remediates the student's misconceptions, and completes a topic. The discourse action begins with the pedagogical level, introduce state, and then it traverses the network and finishes one topic as it reaches the complete state. The tactical level chooses an expository style to implement the pedagogy; question the student, give acknowledgement, or give an answer. The states at this level are refinements of the states at the pedagogic level.

B. Representation of Discourse Strategies. The second important mechanism is the representation of the tutorial strategies in the form of states. The discourse strategies were then extracted from the flow chart and expressed as discourse rules. The rules are written as a frame-like structure using Lisp macro functions, which represent the states in the network. The states are divided into default states and meta states, and each state is further divided into pedagogic and tactical states. Each state consists of a state name and slots. The slots in the default state contain information about
tutoring strategy, text style, and explicit control. The meta states mainly include explicit control mechanism and preconditions. In Figure 19, the execution of the Ask Question state will cause the text generator to generate a question, and then move on to the next default state, Eval_Input. The slots also contain a register to keep track of the completion of the topic, and a flag to update the student model.

C. Control Structure. The discourse control in the network can be divided into a default control structure and a meta control structure. The default control is specified in the default states, so that the tutor moves from one state to another according to a pre-determined path. The meta control abandons the default path and moves to the state that is specified in the meta-rule. The system checks the meta-rules first and if none of the meta-rules fire, then the control flow will follow the default path. This control path is hidden in Figure 16, because the exceptional behavior by the meta-rules can not be predicted in advance. For example, the Eval Input state will be selected right after the Ask Question state as a default path, but the next state is unpredictable, since the student answer could be correct, wrong, or partially correct. This mechanism enables the dynamic behavior of the discourse planner.

Pedagogic Default Rule

(Pedagogic_default *introduce*
(subgoal current-task
update topic-completed
next-state *tutor*))

(Pedagogic_default *tutor*
(subgoal current-task
update topic-completed
next-state *complete*))

Figure 17. The Pedagogic Default Rule

Pedagogic Meta Rule

(Pedagogic_meta *m_tutor*
(precondition topic-completed
prior-state *tutor*
next-state *introduce*))

(Pedagogic_meta *m_complete*
(precondition no-more-topics
prior-state (*introduce* *complete*)
next-state *stop*))

Figure 18. Some Pedagogical Meta Rules

The main disadvantage of earlier discourse management networks [Woolf, 1984; Clancey, 1982] is that they needed to be coupled with some other control mechanism, such as an agenda and an external memory to provide a topic. In
CIRCSIM-TUTOR, since the lesson planner provides a globally coherent lesson plan, the network itself can function solely for delivery purposes while keeping all the advantages of the discourse management network, such as flexible discourse control and explicit representation of discourse strategies.

<table>
<thead>
<tr>
<th>Tactical Default Rule</th>
</tr>
</thead>
</table>
| (Tactical_default)
  | (text-style *ask_question*
  | question
  | content current-task
  | update nil
  | next-state *eval-input*)

<table>
<thead>
<tr>
<th>Tactical Meta Rule</th>
</tr>
</thead>
</table>
| (Tactical_meta)
  | *m_correct*
  | (precondition correct-response
  | prior-state *eval-input*
  | next-state *correct-ack*)

<table>
<thead>
<tr>
<th>Tactical Meta Rule</th>
</tr>
</thead>
</table>
| (Tactical_meta)
  | *m_incorrect*
  | (precondition incorrect-response
  | prior-state *eval-input*
  | next-state *incorrect-ack*)

Figure 19. Tactical Default Rules

Figure 20. Tactical Meta Rules

6.2 Discourse Planning

Discourse planning in CIRCSIM-TUTOR is managed by a simple algorithm. It iterates through the states until a topic becomes complete. Either the student responds with a correct answer or the tutor gives the answer. This section describes important features of the discourse planning.

The Discourse Goal. The discourse planner needs a goal to tutor the student. This goal can be found in the subgoal stack, which the lesson planner has produced. In Figure 13, the subgoals are sequenced by number, so that the discourse planner can carry them out in that order. When the planner finishes carrying out one of the subgoals, it will be removed from the stack, and the planner picks the next one. This cycle continues until the stack is empty, or
is suspended by the plan controller in favor of a student initiative [Woo, 1991c].

Generating Natural Language Sentence. The tactical default states have slots containing information for the text generator. When the planner processes the states, the text-style and content slots will be extracted from the current state. For example, assume that the planner is processing the *ask-question* state (Figure 19), while the text-style slot contains question and the content slot contains the current-task, such as determinant (SV). Binding these two slot values provides us with a logic form, (question (determinant (SV))), which will be passed to the text generator, which generates the sentence, What are the determinants of SV? Then the screen manager will display the sentence in the tutor window.

The logic form may need to be extended to generate richer sentences, since this kind of the logic form only contains information about a particular task or the solution of a problem. The text generator may need to collect more information from many other sources, the domain knowledge base, the student model, the dialog history, and so on.

How to Recognize a Student Initiative. CIRCSIM-TUTOR allows student initiatives during the tutoring session. So the planner must understand whether the student response is a question or an answer by checking the input logic form, which is being passed from the input understander. For example, if the input understander passes a logic form, (answer (determinant SV)(RAP CO)), the first item of the list, answer, indicates that this is an answer. The second item of the list, (determinant SV), is the current topic, and the third item, (RAP CO), is the student answer. Let's assume that the tutor asks the question, What are the determinants of SV? and the student responds with I don't know about SV. Then the input understander recognizes this as an implicit question and returns a logic form, (question (do-not-know) (SV)). The planner receives the logic form and recognizes that this is a student initiative, so it suspends the current plan and carries out the student request; asks the problem solver to get the definition of SV from the knowledge base, and then asks the screen manager to display it.

6.1 Trace of Discourse Transition

Figure 21 shows a short trace of a sequence of discourse transitions. The short arrows represent the pedagogic level transitions; the long arrows represent the tactical level transitions; and the double arrows represent the meta level transitions. The left side of the figure shows the processing of states, and the right side of the figure shows the discourse actions resulting from visiting the states.

The tutor begins by asking a question, then it moves to the evaluate state by the default control rule. At this time, the student responds with a half correct answer, which is recognized by the meta tactical rule, which forces a move to the half-correct state. This state produces an acknowledgement and then another meta rule fires, which recognizes that this is the first try. So the meta rule forces a move to the give-hint state, which produces a hint. Since there is no default and a meta rule applies, the control pops up to the upper level and checks whether the topic has been completed. If not, then control goes back to the introduce state again, and moves down to the tactical level. This time the requestion state is selected, since this is the second try on the same topic.

7. CONCLUSION

7.1 Significant Features

This paper describes the design and development of an instructional planner for a Physiology ITS, CIRCSIM-TUTOR. The planner has several significant features.
## Current Topic: Determinants of SV

- \(\rightarrow, \Rightarrow: \) Pedagogic Level
- \(\rightarrow\rightarrow, \Rightarrow\Rightarrow: \) Tactical Level \( (\rightarrow \text{ Default }, \Rightarrow \text{ Meta } ) \)

<table>
<thead>
<tr>
<th>Discourse States</th>
<th>Discourse Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\rightarrow) INTRODUCE</td>
<td>Tutor: What are the determinants of SV?</td>
</tr>
<tr>
<td>(\rightarrow\rightarrow) Ask-question</td>
<td>Student: RAP and CO</td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Eval-input</td>
<td></td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Meta-tactic3</td>
<td>Tutor: RAP is correct, but CO is not a determinant of SV.</td>
</tr>
<tr>
<td>(Incorrect-one)</td>
<td></td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Half-Correct</td>
<td>Tutor: RAP is correct, but CO is not a determinant of SV.</td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Meta-tactic6</td>
<td>Tutor: Remember. SV is the amount of blood pumped per beat.</td>
</tr>
<tr>
<td>(First-try)</td>
<td></td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Give-hint</td>
<td></td>
</tr>
<tr>
<td>(\rightarrow) TUTOR</td>
<td></td>
</tr>
<tr>
<td>(\Rightarrow\Rightarrow) Meta-pedagogic</td>
<td></td>
</tr>
<tr>
<td>(Not-completed)</td>
<td></td>
</tr>
<tr>
<td>(\rightarrow) INTRODUCE</td>
<td></td>
</tr>
<tr>
<td>(\rightarrow\rightarrow) Requestion</td>
<td>Tutor: What is the other determinant of SV?</td>
</tr>
</tbody>
</table>

Figure 21. Trace of the Discourse Transition Process

First, the planner combines two different instructional planning approaches: lesson planning and discourse planning. Lesson planning produces global lesson plans, which will be carried out during the discourse planning stage. Second, the planner plans dynamically based on the inferred student model; it generates plans, monitors the execution of the plans, and replans when the student interrupts with a question during the tutoring session. Third, the pedagogical knowledge is extracted from the experts and represented explicitly as rules, lesson planning rules and discourse planning rules. The rules are used to generate lesson plans and to control discourse strategies. The system interprets the rules and builds the lesson plans or returns an appropriate discourse action. Fourth, the planner plans at different levels of the hierarchy; the higher level is a simplification or abstraction of the plan (lesson goals) and the lower level is a detailed plan (subgoals), sufficient
to solve the problem. Fifth, the planner allows minimal student initiatives during the tutoring session. If the student asks a question the planner suspends the current plan, carries out the student request, and then resumes the suspended plan.

Since one of the main goals of CIRCSI-TUTOR is to provide a natural language interface, the discourse planner is designed not only to provide sophisticated discourse control, but also to create the internal logic forms for the text generator to generate the sentence. A short tutoring scenario is introduced, which came from a transcript of human tutor and student interaction, to explain the internal process of the system.

7.2 Future Research

The current version of the student model is limited to the overlay strategy, so the planner can support tutoring on the overlay errors only, not the bugs. The tutoring strategy for the bug library has not been developed yet, so the system cannot tutor the student about bugs at the moment.

Another important tutoring strategy is giving a more detailed level hint during the tutoring session. Also it needs more anticipation from other modules; the domain knowledge base needs much more detailed knowledge, the input understander and the text generator need to expand their lexicon and logic forms to contain all the variables at the detailed level, the problem solver needs to be able to access the knowledge base and extract a hint, and the planner needs to have a general strategy for deciding the content of the hint for every situation during the tutoring session.

CIRCSI-TUTOR supports four pre-determined problems as a curriculum, so that it does not really require curriculum planning. Our expert tutors are developing many more procedures for the system, which may require sophisticated curriculum planning in future versions of the system.

8.0 BIBLIOGRAPHY


Charniak, E. and McDermott, D. 1986. Introduction to Artificial Intelligence, Addison-Wesley, Reading, Massachusetts.


Kearsley, G. 1987. Artificial Intelligence and Instruction: Application and Methods, Addison-Wesley, Reading, Massachusetts.


Distribution List

Dr. Philip L. Ackerman
University of Massachusetts
Department of Psychology
75 East Fuss Road
Amherst, MA 01003

Dr. Beth Antonoff
Department of Psychology
University of Massachusetts
Amherst, MA 01003

Dr. David L. Albertson
Code 13H
Navy Personnel R&D Center
San Diego, CA 92152-4888

Dr. Terry Allen
Code 309CE
Office of Naval Research
484 N. Quincy St.
Arlington, VA 22217-3399

Dr. Thomas H. Anderson
Center for the Study of Reading
174 Children's Research Center
15 Cherry Drive
Chapel Hill, NC 27514

Dr. Stephen L. Amick, Chairman
College of Information Studies
Dartmouth College
Hanover, NH 03755

Dr. Gregory Arp
Educational Testing Service
Princeton, NJ 08541

Dr. Michael A. Amundson
HYVAIR
AI Laboratory
520 Westchester Avenue
White Plains, NY 10605

Dr. Pamela A. Baggs
Society of Education
440 W. Wight St., Suite 1300
University of Illinois
Ann Arbor, MI 48104-1230

Dr. Nancy L. Barker
Navy Personnel R&D Center
San Diego, CA 92152-4888

prof. dr. drs. Rens S. Bos
Universiteit van Amsterdam
Documentatie van Militaire documentatie
430 P. Molen- en V. Blume 23
ITALY

Dr. Hans-Joachim Breuerfeld
GMD der Universität
Postfach 4004
D-80899 Bochum, W. GERMANY

Dr. William E. Berry
Director of Life and Environmental Sciences
AP/CHM/ML, P. O. Box 609
Bedford AFB, DC 20123-6449

Dr. Thomas G. Bevery
Department of Psychology
University of Rochester
Rochester, NY 14627

Dr. Kathleen Bierman
Department of Computer Science
San Diego, CA 92136-0209

Dr. Charles C. Chaffin
Table Hall
Department of Psychology
University of Massachusetts

Dr. Charles Chaffin
Table Hall
Department of Psychology
University of Massachusetts

Dr. Charles Chaffin
Table Hall
Department of Psychology
University of Massachusetts

Ambrose, MA 01002

Chief of Naval Education and Training (N-2)
NAP, Pensacola, FL 32508

Dr. Paul Cohn
Purdue University
Engineering Building
W. Lafayette, IN 47907

Office of Naval Research
Code 142
484 K. Grove Street
Arlington, VA 22217-3000

Director
Training Systems Department
Code 15A
Navy Personnel R&D Center
San Diego, CA 92152-4888

Library, Code 238
Navy Personnel R&D Center
San Diego, CA 92152-4888

Naval Training Systems Center
Attn: Dr. Richard N. Cooney, Code 262
1250 Research Parkway
Orlando, FL, 32826-2224

R&D Committee, Army Int. Hq.
Office of the DCMO, MPT, Op. 116-1
Department of the Army, AA 0417
Washington, DC 20360-3000

Communications Officer
Navy Research Laboratory
Code 4227
Washington, DC 20375-5000

Dr. John M. Curnow
Department of Psychology
U.S. Psychology Program
Tulane University
New Orleans, LA 70118

Dr. Michael Curnow
Code 142
Navy Personnel R&D Center
San Diego, CA 92152-4888

Dr. Kenneth R. Cross
Andraus Sciences, Inc.
P.O. Box 119
Santa Barbara, CA 93102

CTM/McDermott Hill Library
2800 Gopher Road
Minneapolis, MN 55440-5288

Dr. Peter C. Cudworth
Director, Center for Cognitive Sciences
380 Ohio State East
1986 Fruit Park, P.O. Box 1680
Columbus, OH 43216-1102

Dr. Charles B. Davis
Information Technology Services
Mall 22, T. S.
Potomac, MD 20854

Margaret Day, Librarian
Applied Sciences Association
P.O. Box 1972
Baton, PA 08008

Dr. Sharon Davy
Florida State University
Department of Psychology
Tallahassee, FL 32306

Dr. J. Stuart Davis
Faculty of Education
University of British Columbia
3521 Main Mall
Vancouver, BC, CANADA V6T 1Z4

Defense Technical Information Center

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. John B. Blaske</td>
<td>Navy Personnel R&amp;D Center</td>
</tr>
<tr>
<td>Dr. Jeffrey Blume</td>
<td>University of California, San Diego</td>
</tr>
<tr>
<td>Dr. Larry D. Byrd</td>
<td>US Office of Personnel Management Office of Personnel Research and Development Corporation</td>
</tr>
<tr>
<td>Dr. samen J. Coles</td>
<td>Army Research Institute, ERDC</td>
</tr>
<tr>
<td>Dr. Harold L. Coles</td>
<td>Fort Leavenworth National Cemetery</td>
</tr>
<tr>
<td>Dr. Dr. A. D. Harden</td>
<td>Army Research Institute, ERDC</td>
</tr>
<tr>
<td>Dr. Richard L. Ferguson</td>
<td>American College Testing P.O. Box 110</td>
</tr>
<tr>
<td>Dr. Robert W. Fike</td>
<td>Department of Psychology, Montgomery College</td>
</tr>
<tr>
<td>Dr. Michael Hildreth</td>
<td>Department of Psychology, University of Richmond</td>
</tr>
<tr>
<td>Dr. David J. Gordon</td>
<td>Center for Adaptive Systems</td>
</tr>
<tr>
<td>Dr. Eliza J. Gordon</td>
<td>Center for Adaptive Systems</td>
</tr>
<tr>
<td>Dr. J. D. Gardner</td>
<td>Department of Defense, Alexandria, VA 22314</td>
</tr>
<tr>
<td>Dr. Louis Gardner</td>
<td>Carnegie-Mellon University, Pittsburgh, PA 15213</td>
</tr>
<tr>
<td>Dr. Carl N. Gardner</td>
<td>Department of Psychology, University of Richmond</td>
</tr>
<tr>
<td>Dr. Adolf Friedman</td>
<td>Department of Psychology, University of Melbourne</td>
</tr>
<tr>
<td>Dr. Merritt F. Green</td>
<td>Department of Cognitive Science</td>
</tr>
<tr>
<td>Dr. Herbert Glanz</td>
<td>Stanford University</td>
</tr>
<tr>
<td>Dr. John H. Gowan</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>Dr. Philip Gowan</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>Dr. David Horgan</td>
<td>University of Pennsylvania, Pittsburgh</td>
</tr>
<tr>
<td>Dr. Mary D. Holsh</td>
<td>M. D. 30th Street</td>
</tr>
<tr>
<td>Dr. Martin J. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. Sue J. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. John H. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. Martin D. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. Richard J. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. Robert J. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
<tr>
<td>Dr. Susan J. Hunter</td>
<td>Department of Psychology, Princeton University</td>
</tr>
</tbody>
</table>