Proc. of the Fourteenth International Florida Artificial Intelligence Research Soc. Conf., Key West, FL, May 2001, pp. 211-215

Using the Atlas Planning Engine to Drive an Intelligent Tutoring System: CIRCSIM-Tutor*Version 3

Bruce Mills
Department of Computer Science
Wisconsin Lutheran College
bruce_mills@wlc.edu

Abstract

CIRCSIM-Tutor Version 3 includes a reactive planner, APE - the Atlas Planning Engine. By using this planner, we are able to effectively respond to dynamic changes in the environment of tutoring students to solve problems in cardiovascular physiology dealing with the regulation of blood pressure. Use of the reactive planner allows us to adapt to student initiatives or unpredictable answers to questions by replacing the planner's responses with newly computed better reactions and allows us to specify new goals to the ITS and have it alter its reaction so as to achieve these new goals. The result is a more cohesive dialog with the student.

Introduction

CIRCSIM-Tutor (Kim 1989) asks the student to reason about the regulation of blood pressure in the human body and the negative feedback loop that acts to keep the blood pressure as constant as possible. In doing so, the ITS helps the student to understand the causal relationships involving seven core physiological parameters shown in Figure 1. Figure 1 also shows the influence of the nervous system, which plays an essential role in blood pressure regulation. (In the diagram, Baro = baroreceptor pressure and NS = nervous system response.) A negative feedback loop, known as the baroreceptor reflex, forces the body's response to a change, or perturbation (such as a hemorrhage or a broken pacemaker) over time. It is divided into three stages: the direct response (DR), the reflex response (RR) and the steady state response (SS). CIRCSIM-Tutor has enough domain knowledge to pose 83 problems or procedures with five levels of complexity to the student. In all of these the student must reason about causal relationships.

* This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014-94-1-0338, to Illinois Institute of Technology. The content does not reflect the position of policy of the government and no official endorsement should be inferred.

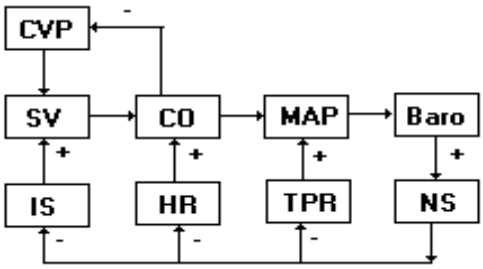

Figure 1: Causal Relationships among Core Variables

CIRCSIM-Tutor begins a session by requesting the student name and then asking the student to choose a procedure from the knowledge base. A procedure begins by describing a perturbation of the cardiovascular system. The ITS then requests that the student predict how the control variables will respond to the perturbation. The student makes predictions by filling in a prediction table (shown in Table 1). As the student makes predictions, the ITS compares the student's answers with correct answers. If the student makes an incorrect prediction, the system begins a natural language tutoring episode.

Variable	**DR**	**RR**	**SS**
Inotropic State	0	-	-
Central Venous Pressure	-	+	-
Stroke Volume	-	-	-
Heart Rate	+	0	+
Cardiac Output	+	-	+
Total Peripheral Resistance	0	-	-
Mean Arterial Pressure	+	-	+

Table 1: Prediction table for heart rate increase due to a broken pacemaker (+: variable increase, -: variable decrease, 0: no change in variable)

CIRCSIM-Tutor version 2 is a dialogue-based intelligent tutoring system (ITS). It conducts a conversation with a student to help the student learn to solve a class of problems in cardiovascular physiology dealing with the regulation of blood pressure. It uses natural language for both input and output, and can handle a variety of syntactic constructions and lexical items, including sentence fragments and misspelled words.

Many students have problems in understanding the baroreceoptor reflex. Rovick and Michael (Rovick and Michael, 1992) believe that the solution for this problem is to have a program that can conduct a dialog with the student (Freedman 1996). CIRCSIM-Tutor version 2 has been shown to be an effective tool in trials with 100 first year medical students at Rush Medical College, but the tutoring dialog that it generates sounds unnatural, repetitious, and sometimes incoherent. In most situations it uses the same tutoring method over and over. With version 3, we vary tutoring methods and provide a more fluent and coherent dialog (Cho 1999).

CIRCSIM-Tutor Version 3 Architecture

The architecture of the new version of CIRCSIM-Tutor is shown in Figure 2. The curriculum planner determines the set of problems that the student may solve. This selection is based on the difficulty of the problem and the success the student has had in solving previous problems.

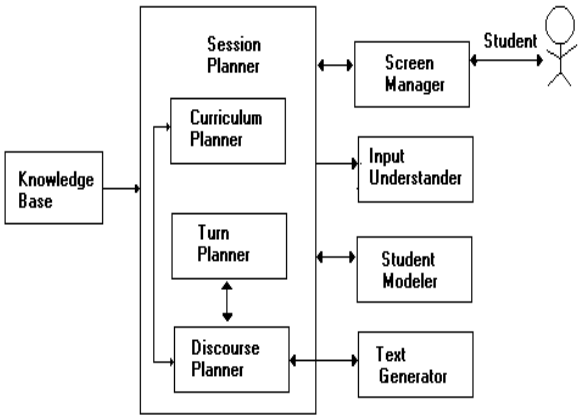

Figure 2 - Architecture of CIRCSIM-Tutor Version 3

The discourse planner is responsible for selecting the tutoring methods. It begins by prompting the student for the first variable in the cardiovascular system affected by this perturbation. The ITS then requests that the student predict the variables in each stage of the body's response. When the student predicts variables incorrectly, those variables become the focus of the tutoring.

The turn planner is still under construction and so at the moment Version 3 is still generating turns a sentence at a time. But one of our goals is to use an opportunistic planning strategy to plan the next tutorial turn and an incremental planning strategy to gather dialog primitives and carry out pedagogical goals.

Finally, the output from the turn planner is processed by the surface sentence generator to display the ITS response.

Wilensky suggests that a planner should have four components: a ***goal detector,*** a ***plan proposer,*** a ***plan projector,*** and a ***plan executor*** (Wilensky 1983**,** Wilensky et al., 1988**).** In CIRCSIM-Tutor the ***goal detector*** acts when it determines that an event has occurred that requires planning, e.g., when the student has requested a procedure, when the student has completed making his tabular response to a directive from the tutor, or when the student responds to a question the tutor has asked.

Our ***plan proposer*** searches a library of plan operators and locates a plan to achieve the goal *did-tutor-DR*. This plan adds two additional goals to the agenda - *did-tutor-primary-variable* and *did-tutor-rest-of-column*. This reflects a need to tutor the student about the identification of the first variable affected by the perturbation (primary variable) and the direction of change and then the need to tutor the student on the other six variables affected by the perturbation and their direction of change.

The ***plan projector*** in CIRCSIM-Tutor stores the plan intentions (a set of goals) on an agenda. Once a goal is on the agenda, the planner searches the operator library to find all the operators whose goal field matches and whose filter and preconditions are satisfied.

Our *plan executor* carries out the sequence of actions described by the plan operators. The executor function takes the next goal from the agenda along with its plan and begins to satisfy each goal by completing activities defined by the goals.

Wilensky also realized that situations arise that require replanning. When our planner realizes that a student has provided a response that is partially correct, a conflict occurs. In this instance, we must replan in order to acknowledge the correct part and tutor the incorrect part of the answer.

CIRCSIM-Tutor is sometimes required to abandon current plans. After making three attempts to solicit the primary variable, we determine that we have failed to achieve the current goal of *tutor-primary-variable*. In this instance, we abandon the current goal and simply replace it with *inform-the-student* of the correct answer. CIRCSIM-Tutor will also abandon plans for tutoring when it determines that the method of tutoring has failed to produce the desired result. For example, we may tutor a variable in the prediction table by a method called *directed line of reasoning*, that is a series of questions that the tutor uses to teach a specific point. If the student does

not give the desired response to this method of tutoring, we will abandon the method and attempt a new method, *tutor-via-determinants*, in which we attempt to teach the student to reason about the determinants (variables which affect another variable) of the incorrect variable and how they affect it.

Planning in CIRCSIM-Tutor Version 3

Planning in CIRCSIM-Tutor is carried out by the Atlas Planning Engine (APE). Atlas is an integrated hierarchical planner that also executes its plans. The benefits include: (1) allowing for reactive planning since it is impossible to account for all possible student responses in a conversation between the ITS and the user; (2) multiple tutoring protocols because human tutors will sometimes change their method of tutoring based on the student answer; (3) multi-turn planning; (4) plan modification and retry; and (5) lexical variety which makes the conversation less repetitive.

The planner is invoked by presenting it with a goal that is placed on an agenda. The planner then searches a user provided operator library to locate matching goals. When it locates a goal for which the preconditions and filters have been satisfied, the initial goal is replaced with the matching goal and the steps or recipe required for satisfying the goal. The original goal is retained, so that if the need arises, a new plan can be made to satisfy the goal (Freedman 2000a, b).

Each operator in the operator library has the following format:

```
def-operator user-supplied-operator-name
:goal  - the goal of this operator
:filter - a list of well-formed formulas that must be in
the database to consider executing this operator
:precond - a list of conditions that must be satisfied
for this operator to run
:recipe -  the steps in achieving the goal of this
operator
:temp  -  temporary conditions in effect for this
operator
```

The planner begins by establishing an overall plan for the tutoring by placing goals on the stack which begin the procedure, correct the procedure, and finally, end the procedure tutoring. Successive goals will form a hierarchy that decomposes the higher level goals into lower level goals. We begin the tutoring by adding four goals to the agenda: *did-start-tutoring*, *did-beginning-of-procedure*, *did-correct-procedure*, and *did-end-of procedure*.

Once the first two goals have been satisfied, the planner locates the operator connected to the goal *did-correct-procedure*. This operator places four additional goals on the stack to tutor the primary variable and the stages of the procedure.

```
(def-operator T-corrects-procedure
  :goal (did-correct-procedure ?student ?procedure)
  :filter  ()
  :precond ()
  :recipe ((goal (did-primary-vbl-section))
        (goal (did-stage dr))
         (goal (did-stage rr))
         (goal (did-stage ss)))
  : temp ())
```

It may be the case that there are multiple ways to satisfy a goal. An instance of this occurs when the student incorrectly identifies the primary variable for the procedure. To accomplish this APE provides a *retry-at* form for the recipe. *Retry-at* allows the planner to choose among several partial plans to satisfy the goal. For the *retry-at* form to be useful, multiple operators must be supplied with the appropriate preconditions enabling the goal to be chosen. Another form available in APE for decision making is the *prune-replace*. The *prune-replace* form pops the top goal off the stack and replaces it with another goal (Freedman 2000a, b).

The following example shows the rules for correcting the primary variable when the student has supplied the incorrect answer. CIRCSIM-Tutor uses the *retry-at* form to give the student two opportunities to choose the correct answer. If the student fails to select the correct primary variable after two attempts, CIRCSIM-Tutor provides the answer.

```
(def-operator primary-vbl-incorrect-1
  :goal (did-correct-primary-vbl)
  :filter  ()
  :precond ((i-input-catg-is incorrect)
      (w-primary-hint-count-is 0))
  :recipe ((goal (did-utter ("First hint")))
      (retract (w-primary-hint-count-is ?x))
      (assert (w-primary-hint-count-is 1))
      (retry-at (w-on-stack did-primary-vbl-instance)))
  :temp    ())
```

Curriculum Planner

The curriculum planner defines an appropriate set of procedures that the student should be asked to solve and then it allows the student to select a procedure from the set. When the student finishes a procedure the curriculum planner constructs a new procedure set on the basis of the student input and tutor's assessment of the

student's ability. The levels of difficulty for the next procedure set can also be adjusted by student input (Cho 1999).

The following operator adjusts the procedure difficulty level up based on the student performance with the previous procedure set. The goal is to adjust the difficulty level (*adjust-diff-level*). APE gives us the opportunity, with the use of preconditions, to make an adjustment by satisfying the changing conditions of the tutoring session. In this case we check the recent performance level, basic adjustment level, a maximum difficulty level, and a current difficulty level. If those preconditions are satisfied, a function to make the adjustment is called and a new value for the difficulty level is produced.

```
(def-operator adjust-difficulty-level-up-cp
   :goal (adjust-diff-level)
   :filter ()
   :precond ((recent-performance-is 2)
         (not(adj-req-is 0))
         (not(current-proc-diff-level-is 5))
         (current-proc-diff-level-is ?old-value)
         (e-increase-one-is ?old-value)
         (the-adjusted-value-is ?new-value))
    :recipe ((retract (no-error
              (current-proc-diff-level-is ?x)))
         (assert (current-proc-diff-level-is
              ?new-value)))
     :temp ())
```

Discourse Planner

The purpose of the discourse planner is choose a method of teaching content, choose a level of interactivity between the student and tutor, and choose a coherent set of sentences to present to the student will teach the concepts. Tutoring one stage of the procedure selected by the student requires the tutor to evaluate the student's tabular predictions for that stage. Within each stage the discourse planner selects the variables to be tutored, the methods by which the variables are tutored, and the topics needed by each method. Because the planner is able to react to the changing circumstances of the tutoring session (e.g., an incorrect or partially correct answer given by the student), the method or topic of the tutoring may be changed as necessary to tutor the misunderstanding or lack of knowledge.

In the example that follows, we find that the student has responded to a question by giving a partially correct answer when asked to give the determinants of a variable. For this goal to be selected we must satisfy the preconditions: the variable (*vbl*) must not be a neural variable, the topic being tutored is the variable's determinants, the method being used is *tutoring determinants*, the level of tutoring is *topic*, and the student model has determined that the answer is only partially correct. If these preconditions are satisfied, then CIRCSIM-Tutor searches its knowledge base to locate the missing determinant. Three goals are added to agenda: acknowledge the part of the answer that was correct (*did-acknowledge-right-part*), give the student a hint about the incorrect determinant (*did-give-hint-of-missing-det*) and finally, ask the student to respond with the correct determinant (*did-ask-the-other-determinant*).

```
(def-operator handle-partial-answer
  :goal (did-handle-partial-answer)
  :filter ()
  :precond ((w-variable-is ?vbl)
     (not (is-neural ?vbl))
     (w-topic-is determinants)
     (w-method-is via-determinants)
     (w-level-is topic)
     (i-input-concept-is ?partial)
     (e-get-missing-determinant ?vbl ?partial ?det))
  :recipe  ((retract (no-error (w-partial-answer-is ?x)))
     (assert (w-partial-answer-is ?partial))
     (goal(did-acknowledge-right-part ?vbl ?partial))
     (goal (did-give-hint-of-missing-det ?vbl ?det))
     (goal (did-ask-the-other-determinant ?vbl ?det)))
  :temp    ())
```

Student Modeler

The focus of the student modeler is to assess the correctness of the student answer. For each student answer the student modeler provides the planner with a correctness category and information about the incorrect part of the answer. Correctness is classified into one of eight categories: correct, partially correct answers in which some part of the answer is correct and the rest is incorrect; incorrect answers, misconception answers that show a common confusion about the concept being tutored; near miss answers in which the answer is reasonable but does not refer to one of the variables on the prediction table; grain-of-truth answers in which the student provides an answer but not the answer to the tutor's question; 'I don't know' answers; other incorrect answers; mixed answers that are a combination of answers from other categories (Zhou et al., 1999).

Once the student responds to a question, the student modeler categorizes the answer and prepares a performance evaluation. This evaluation includes an assessment of the student competence, a student reply history that contains information about the causal chain being tutored, the actual student responses to questions, a record of the number of errors the student made while solving the problem, and a tutoring history containing

tutoring strategies and topics (Zhou et al., 1999).

Turn Planner

The input to the turn planner consists of a sequence of utterances from the discourse planner. These utterances were possibly generated by different parts of the plan (e.g., an acknowledgment followed by a hint followed by a new question).

The purpose of the turn planner (still under implementation) is to make the discourse more fluent and comprehensible. To accomplish this goal the turn planner will insert discourse markers that make clear the logical relationship between two sentences. It chooses appropriate acknowledgements to make clear the correctness of the student response (e.g., "Yes", "Great", "Right"). The turn planner will make the decision as to when variable names are to be abbreviated (e.g., "CO" rather than "cardiac output") and also will use various softeners in the discourse process (e.g., "Can you tell me, what is …." rather than "What is …"). (Yang et al., 2000)

Summary

CIRCSIM-Tutor Version 3 is a natural language-based ITS that uses a reactive planner to tutor students in the domain of reflex control of blood pressure. The advantage of using the reactive planner is the ability to replan an entire tutoring session or a small part of the session based on the student's response to a question posed by the ITS.

Acknowledgement

I wish to thank Reva Freedman for providing the Atlas Planning Engine and for her work which inspired the architecture of CIRCSIM-Tutor Version 3.

References

Cho, B. (1999), "A Curriculum Planning Model for an Intelligent Tutoring System", *Proceedings of the Twelfth Florida Artificial Intelligence Symposium (FLAIRS-99)*, pp. 1-5

Freedman, R. (1996), *Interaction of Discourse Planning, Instructional Planning and Dialogue in an Interactive Tutoring System*, Ph. D. Dissertation, Northwestern University

Freedman, R. (2000a), "Using a Reactive Planner as the Basis for a Dialog Agent", *Proceedings of the Thirteenth Florida Artificial Intelligence Symposium (FLAIRS-2000)*, pp.203-208

Freedman, R. (2000b), "Plan-based Dialog Management in a Physics Tutor", *Proceedings of the Sixth Applied Natural Language Processing Conference*, Seattle, WA.

Kim, N. (1989), *CIRCSIM-Tutor: An Intelligent Tutoring System for Circulatory Physiology*, Ph. D. Dissertation, Illinois Institute of Technology

Rovick, A., Michael, J. (1992), "The Predictions Table: A Tool for Assessing Student's Knowledge", *The American Journal of Physiology*, 263 (6, part 3): S33-S36

Wilensky, R. (1983), *Planning and Understanding: a Computational Approach to Human Reasoning*, Addison-Wesley, Reading, MA

Wilensky, R., Chin, D., Luria, M., Martin, J., Mayfield, J., & Wu, D. (1988). "The Berkeley Unix Consultant Project", *Computational Linguistics*, 14(4), pp. 35-84

Yang, F-J., Kim, J. H., Glass, M., Evens, M. (2000), "Turn Planning in CIRCSIM-Tutor", *Proceedings of the Thirteenth Florida Artificial Intelligence Symposium (FLAIRS-2000)*, pp. 60-64

Zhou, Y., Freedman, R., Glass, M., Michael, J., Rovick, R. Evens, M. (1999), "What Should the Tutor Do When the Student Cannot Answer a Question?", *Proceedings of the Twelfth Florida Artificial Intelligence Symposium (FLAIRS-99)*, pp. 1-5