A Simulation-Based Tutor that Reasons about Multiple Agents

Christopher Rhodes Eliot III and Beverly Park Woolf

Department of Computer Science
University of Massachusetts, Amherst
Amherst, Ma. 01003
{Eliot, Bev @cs.umass.edu}

Abstract

This paper examines the problem of modeling multiple agents within an intelligent simulation-based tutor. Multiple agent and planning technology were used to enable the system to critique a human agent's reasoning about multiple agents. This perspective arises naturally whenever a student must learn to lead and coordinate a team of people. The system dynamically selected teaching goals, instantiated plans and modeled the student and the domain as it monitored the student's progress. The tutor provides one of the first complete integrations of a real-time simulation with knowledge-based reasoning. Other novel techniques of the system are reported, such as common-sense reasoning about plans, reasoning about protocol mechanisms, and using a real-time simulation for training.

Introduction

Simulation-based tutoring provides students with valuable practice, which is often impossible to obtain in the real world. A simulation coupled with an intelligent tutor can be particularly effective because students can practice problem-solving in a realistic setting while the tutor can recognize when situations arise that are well suited for targeting specific learning activities. Properly situated teaching is effective because the student can comprehend the significance of knowledge immediately applied to solve a current problem. We built a simulation-based tutor incorporating these ideas for teaching Advanced Cardiac Life Support (ACLS) to medical personnel.

In the Cardiac Tutor the student was required to supervise several independent assistants. Each assistant performed tasks critical to the procedure and these agents were coordinated by the student team leader. Monitoring the student required reasoning about the roles and state of all agents in order to evaluate student actions. This problem differs from distributed reasoning because tasks are distributed but reasoning is not. All problem-solving and coordination was centralized as the student team leader reasoned about multiple-agents executing tasks in parallel.

The problem is significant because it appears whenever people must be trained to coordinate working teams. The tutor was designed to meet the practical needs of medical personnel, and was used by students training in ACLS. The mechanism was implemented within a complete tutor and partially validated during formative studies of the system.

The Cardiac Tutor integrated a number of advanced technologies, including knowledge-based tutoring, simulation, plan recognition, student modeling, domain reasoning and multimedia. Integration of these diverse elements is one of the novel aspects of this system. The user model for the Cardiac Tutor has been described elsewhere in [Eliot and Woolf, 1994, 1995] and the iterative development and evaluation process are described in [Eliot and Woolf, 1996].

This paper presents an overview of the system and then focuses on the planning mechanism showing how it reasons about multiple agents. The description of the planning mechanism includes examples of how it works, its relation to a transition network representation, and its use within the larger system. Novel aspects of this technique include meta-level reasoning about multiple agents, partial matching of student actions, using a linear formalism to efficiently encode parallel actions with support for multiple roles, subprotocols, conditional steps and synchronization.

An Overview of The Cardiac Tutor

The Cardiac Tutor instructs medical professionals to lead Advanced Cardiac Life Support (ACLS) teams [AMA, 1992; Cummins, 1994]. Proper training for ACLS leadership requires approximately two years of closely supervised clinical experience. Computerized instruction has the potential to significantly reduce the cost of this training.

Instruction in the Cardiac Tutor was based on a realistic simulation of a cardiac resuscitation situation allowing the student to practice while receiving knowledge-based feedback. The mechanisms for simulation-based tutoring were broken down into four fundamental components:

- Goal selection: reasoning about the student's needs and the best context for meeting them.
- Plan formation: moving from one context to another based on goal satisfaction.

• Plan instantiation: simulating a new context/plan, even when this included the revision or abandonment of outstanding plans.

• Situated reasoning: applying situation-specific knowledge to help the student achieve learning goals when the system reached a desired context.

Figure 1 shows a time-varying trace of the integrated simulation, planner, plan recognition system, and student-model reflecting the actions of the student, the system and their effects on each other. The bottom line, Patient, represents simulated clinical reality. The user actions, Student, could be inconsistent with the desired behavior, Expert. To detect such inconsistencies, a plan recognition phase, Planner, compared expert and student actions. Because the instructional feedback, Tutor, reflected the current context of the simulation model, the recommendations were robust.

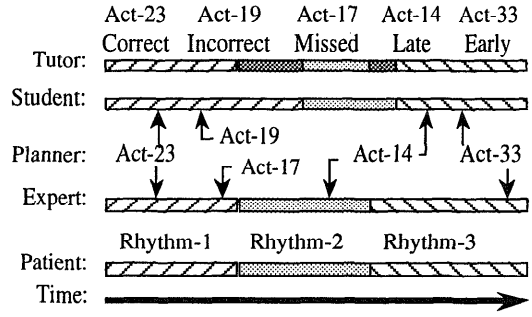

Figure 1. Functionality of the Simulation and Planning Mechanisms.

The Student's View of the Tutor

The simulated patient, Figure 2, underwent simulated cardiac arrest, while the student employed a variety of interventions, including defibrillation, intubation, IV, or medications. The patient's clinical state and ECG were depicted as a realistic real-time multimedia presentation.

Figures 2 and 3 show the simulation of a patient undergoing a series of arrythmias. Figure 2 shows five separate actions being selected by the student using pop-up menus. The student later tried a sequence of drugs but the ECG degenerated to ventricular fibrillation, Figure 3. The student had to recognize the changed patient condition and select the appropriate treatment.

Following each simulation the recorded history of the student's actions was made available for review, providing retrospective feedback. Each action in the history was connected to records of the simulation state and the knowledge base, so the student could re-examine his decisions and their context in detail.

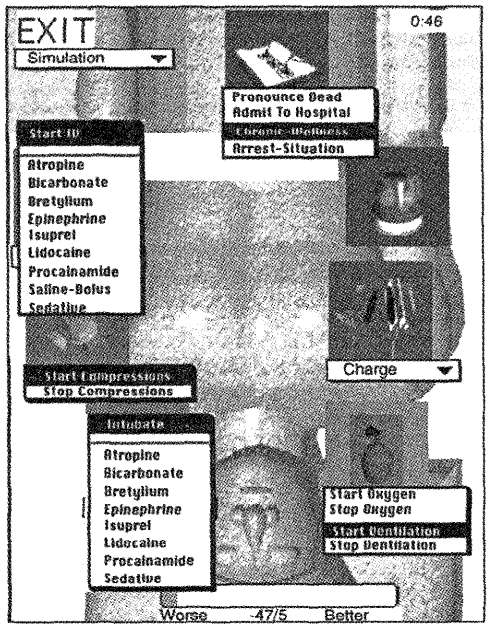

Figure 2. The Simulated Patient

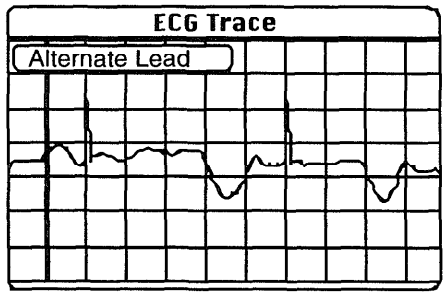

Figure 3. Simulated ECG Trace.

Several important observations of the system were made during formative evaluations. These studies involved two classes of fourth year medical students. The computerized training appeared to be equivalent to time spent with a human instructor during traditional training. Students perceived the cardiac simulation as it was intended, i.e., clinical interventions applied to simulated patients undergoing cardiac arrest. The same cognitive skills were required to operate the simulation as required during traditional training. We have yet to obtain data providing definitive support for each of these claims, as explained in [Eliot and Woolf, 1996], but evidence supports the claim that ongoing practice using the Cardiac Tutor is an effective way to maintain medical skills and that group practice using the Cardiac Tutor was particularly effective for creating a positive learning environment.

Prior Research

Early results with medical reasoning systems provide the clearest indication of how difficult is the presentation of coherent explanations [Szolovits, 1979]. Canned-text and simple translations of execution traces do not provide high quality explanations. Work on knowledge representation and reasoning [Rissland, 1978; de Kleer, 1979; Patil, 1981] illustrate that multi-dimensional representations are needed for full use of a system's knowledge. Current explanation systems attempt to present "correct" information within context based on understanding a user's immediate knowledge and learning needs [Suthers et al., 1992].

Attempts have been made to make simulations give causal explanations in addition to merely generating descriptions of a system's behavior [Kuipers, 1986; Amador et al., 1993]. For example, Amador et al. provide techniques for obtaining explanations from equations and numeric simulations. Currently, few intelligent tutors can modify a simulation in order to customize their curriculum to the user's needs, but see [Katz et al., 1992].

User and student modeling techniques focus on stereotypes [Kay, 1994; Orwant, 1994]; plan recognition; bug libraries [Van Lehn, 1988] and overlay models [Suthers et al., 1992]. Plan recognition normally is based upon goal decomposition [Broverman, 1991]; our system instead is based upon linear protocols.

Distributed AI work focuses on communication of agents and techniques that allow agents to create plans semi-independently to achieve a single global goal [Neiman et al., 1994; Kuokka et al., 1994; Durfee et al., 1985]. Our system describes actions of multiple agents, but avoids the problem of plan coordination because we are training a single human agent to perform the leadership role. Because planning and problem-solving are centralized in our system there is no need to account for separately constructed sub-plans.

Multiple Agents in a Simulation-Based Tutoring System

The plan recognition system served as a model of the domain for comparison with the student's actions. In the simulation-based tutor, the student performed actions which were evaluated by the tutor to assess consistency with the expert's actions, encoded as a planning formalism.

Medical domain experts often express their knowledge in a different form than knowledge engineers would like, typically describing examples or linear sequences of actions, rather than giving general principles, rules or plans. The domain model consisted of a set of protocols integrated with a real-time simulation of cooperating agents following the student's orders. The plan recognition system used a protocol formalism designed to approximate the conceptual structure of domain experts communicating with each other. Therefore, the protocol interpreter differed from most planning formalisms as it was based upon sequential protocols, not goals, and protocol selection was determined by a straightforward mapping from the current state of the simulation. Plans were expressed in a linear form and the system implicitly recognized opportunities for parallel activity. The recognition system supported recovery from incorrect user actions and accounted for synchronization of multiple agents.

The tutor provided students with the opportunity to lead simulated resuscitations. The tutoring system reasoned about the student, both as an agent within the simulation and as a student learning a task. The student was responsible for directing the simulated agents and most student commands were simulated as orders for simulated assistants to perform medical tasks. In addition, the student was represented within the simulation and directly performed some actions, such as operating the simulated defibrillator. The simulated team leader shouted orders for another agent to perform when the student selected a command from a menu. Each order was simulated as realistically as possible, including an appropriate real-time delay as the task was performed.

The domain required complex representations and reasoning mechanisms. Many features had to be generalized for multiple instances, such as:

- Multiple agents, including doctors, nurses, EMTs.
- Multiple roles for each agent, such as airways, medications.
- Multiple actions for each agent, such as start IV, intubate, defibrillate.
- Multiple orders, since actions could be done out of order due to parallel activity.

The Protocol Interpreter

The system reasoned about the student's ability, by comparing student actions with a model of an expert's behavior. Any action or order expected of the team leader was accepted as a valid student action. Any other action was considered incorrect. Because multiple agents were represented and could act in parallel there was generally more than one correct action that the student could select. Any action that any agent was ready to perform was a valid action for the student to request, either a command for a secondary agent or an action to perform directly. Figure 4 shows a simplified version of the protocol interpreter. The correct protocol had to be selected initially by analyzing the state of the simulation. In this example, Protocol-3 was selected so Protocol-1 and Protocol-2 remained inactive. If the simulation had been in some other state initially then Protocol-1 or Protocol-2 could have been selected.

Once a protocol was selected, the interpreter normally followed it to the end, provided that the simulation remained within a specified range of states. If the simulation state ever satisfied a termination predicate associated with the current protocol then that protocol was terminated and the selection process was restarted. In this

example, the student had already completed the first action, namely, Act-31 so the next action required was to perform Act-32; any other action was incorrect. The current recommendation was indicated by a pointer which marks the last completed action in the selected protocol.

Encoding the knowledge in this domain required a more sophisticated representation than depicted in Figure 4. Some actions were optional in some situations. For instance, if Act-32 was optional then the current recommendation would be the set including actions: R = {Act-32, Act-33}. A student action, A, was correct if A was a member of R. Since several actions in sequence could be optional the set of current recommendations could include many actions. If the student's actions were always correct, updating the protocol pointer was comparatively straightforward. However, incorrect student actions were allowed to affect the state of the simulation, possibly resulting in movement to a state where currently recommended actions were impossible or meaningless [Broverman, 1991]. The protocol interpreter required additional planning knowledge to detect and correct such problems.

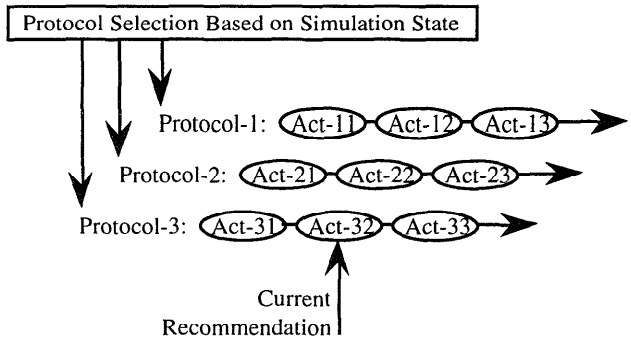

Figure 4. Protocol Recognition.

The knowledge base allowed preconditions and postconditions to be specified for protocol actions. When the protocol interpreter detected an incorrect student action it notified the student, then examined the preconditions of actions in the current set of recommendations and then skipped impossible actions by moving the pointer forward.

The algorithms used in this plan recognition system have not been formally analyzed, but they avoid unbounded search. The linear protocol representation supported efficient reasoning about all possible combinations of assumptions regarding the student's choice of optional actions because incorrect actions could be skipped but not reordered.

An Example

To illustrate the various steps in the protocol and planning mechanisms, we provide diagrams showing the state of the plan recognition mechanism during a sequence of states involved in tracking a student's actions during a simulation.

The protocol interpreter modeled multiple agents acting in parallel. Each action in a protocol was assigned to a role, by the domain expert, and each agent was dynamically assigned one or more roles by the system. In the cardiac domain the roles are leader (responsible for issuing orders and operating the defibrillator), airways (responsible for ventilation and intubation), circulation (chest compressions) and medications (in charge of medications, IV lines and drugs). Every agent interpreted the protocol semi-independently, selecting actions appropriate to its assigned roles and skipping others.

The first recommended action in this example is to charge the defibrillator to 200 joules. After protocol selection was complete and the plan recognition system fully initialized, the nine step universal ventricular fibrillation/ventricular tachycardia (VF/VT) protocol was selected and two subprotocols were activated. The first step of the VF/VT protocol activated the subprotocol for a sequential defibrillation sequence at 200 joules, which in turn activated the subprotocol for a stand clear sequence, Figure 5.

The simulated team leader agent is identified as 'Lead', and the other managers are named 'Air' for airways, 'Circ' for circulation and 'Meds' for medications. All protocol pointers for the four agents initially pointed to the first step in the VF/VT plan. Each protocol pointer was moved forward to the first plan applicable to a role assigned to that agent. When subprotocol steps were encountered, the subprotocol was instantiated, the protocol pointer made to point to the first step in the subprotocol and then the pointer was moved forward linearly.

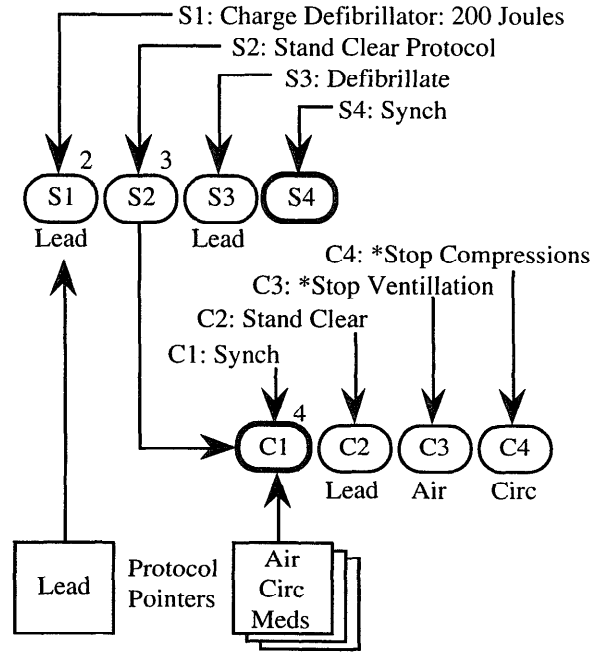

Figure 5. Sequential Defibrillation Procedure.

The protocol pointer for the team leader first pointed to a subprotocol step, so the sequential defibrillate subprotocol shown in Figure 5 was instantiated and the protocol pointer was made to point to the first step in that subprotocol, plan step S1. This step was assigned to the team leader role, so the protocol pointer for the team leader agent was not advanced further. Hence the first (and in this case only) recommended action was to charge the defibrillator to 200 joules. The other protocol pointers were initialized and advanced similarly, except that none of the agents other than the team leader could charge the defibrillator. Hence, those protocol pointers moved past step S1 to step S2. This was another subprotocol step so the stand clear subprotocol was activated and the three agents assigned to medications, airways and circulation eventually pointed to the first step in this subprotocol, C1.

Step C1 was a synchronization step, with sequence number 4. As each protocol pointer was moved to this step another check was performed to determine if the synchronization condition was satisfied on all protocol pointers for all agents. In the state depicted in Figure 5, the protocol pointer for the team leader was located at step S1 which had sequence number 2 which was less than the sequence number for the synchronization step C1 (i.e., 2 < 4), so the synchronization condition was not satisfied. Since the synchronization was not complete the agents assigned to airways, circulation and medications had to wait for the team leader to charge the defibrillator. Those three agents were not recommended to perform any action in the state shown.

This example continues until all the action items are complete.

Actions and Synchronization

Actions are not as simple as suggested above. In the following discussion we replace specific actions with the generic 'Act-nn' to explain actions and synchronization in detail.

The role assigned to each action was part of the static knowledge about actions. The assignment of roles to agents was automatically defined at the beginning of each simulation depending upon how many helper agents were available. The number of helpers and their role assignments did not change during a simulation. The simulated team leader was controlled by the student and assigned the single role of leader. The other roles were distributed among the available helper agents as equally as possible.

The team leader was responsible for directing all the agents; hence user commands were interpreted as orders carried out by agents. Each agent was assigned one or more roles and each action was assigned a specific role. The key to plan recognition using protocols is the computation of the set of actions that are currently recommended, Figure 6. In this example actions Act-31 and Act-35 are recommended; other actions are incorrect. Each agent had a separate protocol pointer and interpreted the protocol semi-independently, selecting actions belonging to its assigned roles while skipping other actions. Synchronization steps served to coordinate agents. Any agent reaching a synchronization step paused, waiting for the other agents to catch up, then all agents continued.

Initially the protocol pointer for each agent was set to the start of the selected protocol. Then the pointer was advanced past any actions that were irrelevant to the roles assigned to the associated agent. The set of actions recommended for any agent included the current steps for each role assignment. If a recommended action was optional then the following step was also recommended. A sequence of several actions could all be in the current set of recommended actions in this way.

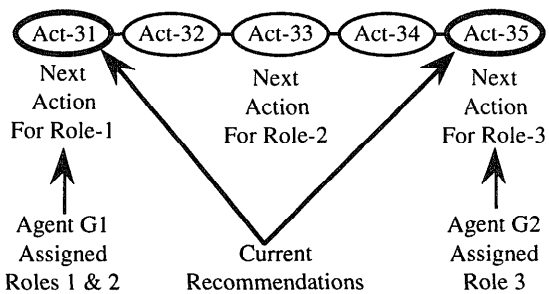

Figure 6. Multiple Agent Protocols.

The set of recommended actions included the action following the last action completed and any subsequent actions if that action was optional. Synchronization steps could be optional, but determining this required reasoning about all the agents. A copy of all of the protocol pointers was created as the first step in determining the recommended actions. These duplicate pointers were then advanced as far as possible past all optional steps to determine an upper bound. A second scan of the protocol for each agent was then done to collect the actual recommended actions using the previously determined upper bound to limit the search. This computation accounted for all possible combinations of assumptions about which optional steps the student might elect to perform.

Figure 7 shows how a synchronization action was interpreted. Action steps A1 and A4 had to be performed by the agent assigned role R1, which in this case was agent G1 and, similarly, action steps A2 and A3 had to be performed by agent G2 who was assigned role R2. When action A1 was performed agent G1 was updated, but had to wait at the synchronization step. When action A2 was complete then both agents G1 and G2 completed the synchronization step.

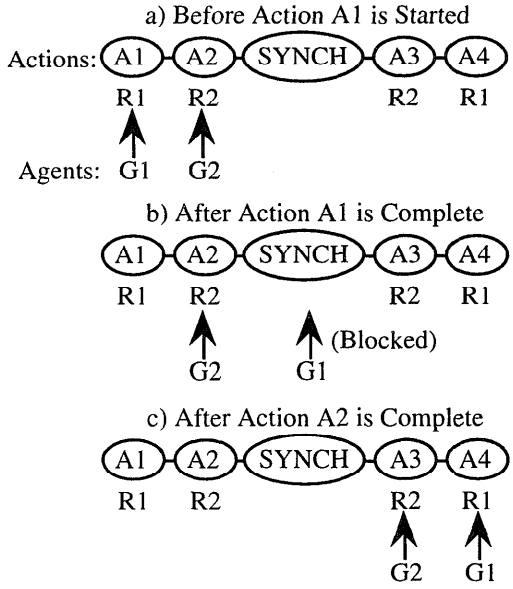

Figure 7. Synchronization of Agents.

Optional actions could interact with synchronization steps, Figure 8. The recommended actions, as specified by the protocols are indicated by the lines forming inverted trees. Several steps before the synchronization step were optional. The analysis to determine the recommended actions requires several steps:

- Agent G1 could skip actions A1 and A3, and was not responsible for action A2, so agent G1 can reach the synchronization step.
- Agent G2 could skip action A2 and was not responsible for actions A1 and A3, so agent G2 can also reach the synchronization step.
- Since both agents can reach the synchronization step it was also optional and both agents may perform later actions.
- Since A4 was optional, agent G2 could next perform any of the actions {A2, A4, A7}.
- Since A5 was required, agent G1 could next perform any of the actions {A1, A3, A5} but not A6.

In this state the recommended action set was {A1, A2, A3, A4, A5, A7}. Action A6 was not recommended because it must be performed by agent G1 who must complete action A5 first in every allowed sequence of actions.

Suppose that the student selected action A4 next. Because action A4 was recommended on the assumption that the synchronization step was satisfied, the student must have elected not to perform any of the steps A1, A2 and A3. Consequently, selecting action A4, which was performed by agent G2, affects the set of actions which agent G1 may next perform; actions A1 and A3 are no longer allowed because the sequences <A4, A1> and <A4, A3> did not satisfy the synchronization step before action A4 was taken. Hence, following action A4 the recommended action set consisted of {A5, A7}.

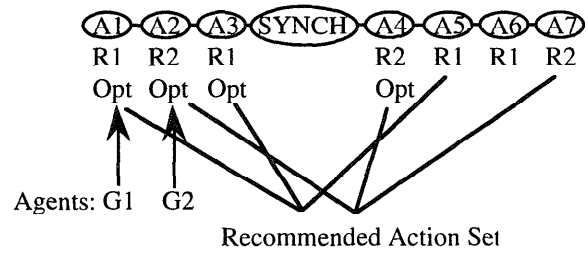

Figure 8. Optional Actions and Synchronization.

Transition Networks

Medical experts presented domain knowledge in a format very similar to the linear protocol representation of Figures 4 to 8. We chose to directly implement knowledge in this form so that changes suggested by the domain experts could be easily incorporated into the system. However, the underlying semantics of the protocols may be more clearly understood by observing how they may be transformed into transition networks, Figure 9.

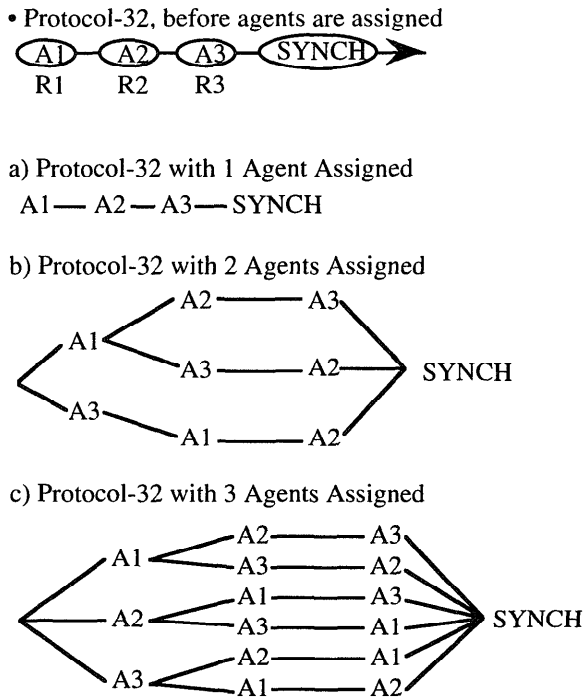

Figure 9 Transition Networks Induced by a Protocol.

The shape of the transition net varies depending upon how many agents are available for parallel activity. Our system did not explicitly perform this transformation; the

plan recognition process implicitly implemented equivalent semantics. In Figure 9 the nodes are labeled with the action leading from the previous state into the node.

Several additional mechanisms were implemented which improved the expressive power of the formalism and made it easier to use. Conditional actions extended the concept of optional actions and were used to restrict protocol actions to certain simulation states. Many actions require this context sensitivity, such as: Start-IV, unless the IV is already in place or Charge-Defibrillator, when defibrillator is not charged.

Conclusions

The tutor implemented mechanisms for goal selection, plan formation, and plan instantiation within situated contexts. It included an accurate descriptive model of the emergency room environment and general patient status, combined with a causal model of cardiac function and related physiologic systems. The tutor consisted of:

- a simulation, representing the problem-solving environment;
- a student model, to guide the learning process;
- a bias mechanism, making the simulation adaptive; and
- a plan recognition system, for constructing the student model.

Multiple agent and planning technology enabled the Cardiac Tutor, unlike typical teaching systems, to go beyond simple classification of student actions as correct or incorrect by specifying how an incorrect user action related to the expert action.

Dynamic construction of the student model involved monitoring student actions during the simulations and evaluating these actions in comparison with an expert model encoded as a multi-agent plan. The plan recognition techniques are novel and allowed the expert knowledge to be expressed in a form that is natural for domain experts. The multi-agent reasoning used in this system differs from most distributed AI reasoning because the system reasoned about a human agent reasoning about multiple agents. Consequently, the difficult issues involving combinations of separately developed partial plans does not arise in this research. Issues of reasoning about multiple agents, using a real-time simulation for training, and reasoning about protocol mechanisms were addressed.

References

Amador, F., Finkelstein, A., and Weld, D. 1993. Real-Time Self-Explanatory Simulation. In Proceedings of AAAI-93, 562-567. Washington D. C.

American Heart Association, 1987. *Textbook of Advanced Cardiac Life Support* (second edition). Dallas, Texas.

American Medical Association, 1992. Journal of the American Medical Association 268 (16).

Broverman, C. 1991. Constructive Interpretation of Human-Generated Exceptions During Plan Executions. Ph.D. diss., Technical Report 91-9, Computer Science Dept., University of Massachusetts, Amherst.

Cummins, R. O. ed. 1994. *Textbook of Advanced Cardiac Life Support.* Dallas, Texas: American Heart Association.

de Kleer, J. 1979. Causal and Teleological Reasoning in Circuit Recognition. MIT-AI-TR 529, Cambridge, Mass.

Durfee, E. H., Lesser, V. R., and Corkill, D. D. 1985. Increasing Coherence in a Distributed Problem-Solving Network, In Proceedings of IJCAI-85, 1025-1030.

Eliot, C and Woolf, B. 1994. Reasoning about the User within a Simulation-based Real-time Training System. In Proceedings of the Fourth International Conference on User Modeling, 121-126.

Eliot, C. and Woolf, B. P. 1995. An Adaptive Student Centered Curriculum for an Intelligent Training System. *User Modeling and User-Adapted Interaction* 5: 67-86.

Eliot, C. and Woolf, B. P. 1996. Iterative Development and Validation of a Simulation-based Medical Tutor. Third International Conference on Intelligent Tutoring Systems, University of Montreal. June 1996.

Katz, S., Lesgold, A., Eggan, G., and Gordin, M. 1992. Modeling the Student in Sherlock II. *AI and Education* 3 (4).

Kay, J. 1994. Lies, Damned Lies and Stereotypes: Pragmatic Approximations of Users. In Proceedings of the Fourth International Conference on User Modeling 175-184.

Kuipers, B. J. 1986. Qualitative Simulation. *Artificial Intelligence* 29 (3): 289-338.

Kuokka, D., and Livezey, B. 1994. A Collaborative Parametric Design Agent. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 387-393.

Neiman, D. E., Hildum, D. W., Lesser, V. R. and Sandholm, T. W. 1994. Exploiting Meta-Level Information in a Distributed Scheduling System. In Proceedings of the Twelfth National Conference on Artificial Intelligent, 394-400.

Orwant, J. 1994. Apprising the User of User Models: Interface Guidelines. In Proceedings of the Fourth International Conference on User Modeling, 151-156.

Patil, R. S. 1981. Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis. Ph.D. diss., MIT/LCS/TR-267, Cambridge, Mass.

Rissland, E. (Michener). 1978. Understanding Understanding Mathematics, Cognitive Science 2 (4).

Suthers, D. D., Woolf, B. P. and Cornell, M. 1982. Steps from Explanation Planning to Model Construction Dialogues. In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, 24-30.

Szolovits, P. 1979. Artificial Intelligence and Clinical Problem Solving. MIT/LCS/TM-140.

Van Lehn, K. 1988. Student Modeling. In *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Erlbaum.