Situated Plan Attribution for Intelligent Tutoring

Randall W. Hill, Jr.

Jet Propulsion Laboratory / Caltech
4800 Oak Grove Drive M/S 525-3660
Pasadena, CA 91109-8099
hill@negev.jpl.nasa.gov

W. Lewis Johnson

USC / Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
johnson@isi.edu

Abstract

Plan recognition techniques frequently make rigid assumptions about the student's plans, and invest substantial effort to infer unobservable properties of the student. The pedagogical benefits of plan recognition analysis are not always obvious. We claim that these difficulties can be overcome if greater attention is paid to the situational context of the student's activity and the pedagogical tasks which plan recognition is intended to support. This paper describes an approach to plan recognition called situated plan attribution that takes these factors into account. It devotes varying amounts of effort to the interpretation process, focusing the greatest effort on interpreting impasse points, i.e., points where the student encounters some difficulty completing the task. This approach has been implemented and evaluated in the context of the REACT tutor, a trainer for Operators of deep space communications stations.

Introduction

Plan recognition and agent modeling capabilities are valuable for intelligent tutoring (Corbett et al., 1990; Johnson, 1986), as well as other areas such as natural language processing (Charniak&Goldman, 1991), expert consultation (Calistri, 1990), and tactical decision making (Azarewicz et al.,, 1986). However, such capabilities are difficult to implement and employ effectively, for the following reasons. Plan recognition techniques can be *rigid*--they assume the agent is following a known plan step by step, and have difficulty interpreting deviations from the plan. The modeling process can be *underconstrained*, postulating mental activities that are difficult to infer from the agent's observable actions. An example of this style of modeling can be seen in (Ward, 1991), where the tutor attempts to track the student by generating production paths that could have led to an observed action. Finally, they tend to be *unfocused*---they do not target their analysis on those situations where tutorial intervention is warranted. For instance, intelligent tutors that use model tracing (Anderson et al., 1990) to interpret student actions tend to intervene whenever the student wanders off of a correct solution path; this intervention policy is potentially disruptive and does not appear to be based on an analysis of whether it is appropriate to intervene.

This paper describes an approach to plan recognition called *situated plan attribution* that takes these factors into account. Situated plan attribution analyzes both the student's actions and the environmental situation. Attention to the situation is important because it allows the plan recognizer to recognize when the student must deviate from the usual plan, as well as alternative ways of achieving the goals of the plan. This flexibility avoids the rigidity problems of other techniques such as Kautz and Allen's deductive approach (Kautz&Allen, 1986), which assumes that all possible ways of performing an action are known, and every action is a step in a known plan.

Motivation for Approach

The objective of situated plan attribution is to inform and guide the tutoring process. We believe that plan recognition systems can and should be optimized to support their intended use. Accordingly, our technique applies greatest analysis effort to interpreting situations where the student might benefit from interactions with the tutoring system. Less effort is devoted to plan recognition when tutorial interaction is not justified on pedagogical grounds. Our stance is consistent with that of (Self, 1990), who argues that to make student modeling tractable one must focus on realistic, useful objectives.

These tutorial interaction points are known as *impasse points*. An impasse is defined in this work to be an obstacle to problem solving that results from either a lack of knowledge or from incorrect knowledge (Hill, 1993; Brown&VanLehn, 1980; VanLehn, 1982, 1983). Cognitive modeling studies suggest that such impasse points are natural learning opportunities (Hill, 1993; VanLehn, 1988; Hill&Johnson, 1993a,b). When the student is at an impasse, he or she naturally seeks information that can be used to overcome the impasse and continue the task. Information offered by the tutor at such points is readily accepted and assimilated. A tutor that is sensitive to such impasses does not run the risk of annoying the student with interruptions--the student's problem solving has already been interrupted by the impasse. The tutoring system need not intervene in a heavy-handed fashion; it can serve as an information resource that the student can turn to for

assistance as needed. The student therefore has a greater sense of control over how the task is performed.

Implementation of the Approach

Situated plan attribution has been implemented and evaluated in the context of the REACT tutor, a trainer for Operators of deep space communications centers. REACT monitors trainees while they operate a set of complex, interactive devices. There are three entities in REACT's tutoring domain: the tutor, the student, and a simulation of the environment (i.e., the devices). The student is assumed to have some understanding of operational procedures. However, the devices may be in unexpected states or behave in unexpected ways; the student must learn to recognize such situations and deviate from the standard procedures as necessary. REACT recognizes when the student has reached an impasse, because the student's action has failed or cannot achieve its intended purpose in the devices' current state. It then coaches the student through the impasse.

REACT's plan recognition capability is not rigid because it has knowledge of device states and actions that affect them, as well as knowledge of plans. It avoids excessive underdetermined student modeling because it focuses on observable student actions and their effects. REACT generally does not intervene with the student unless a student has already received an error message from the device. As long as the student's overall plan is appropriate, interaction centers on the device errors and how to correct them. When necessary it employs an expert cognitive model to determine what action an expert would take in a given situation. Otherwise a weaker, recognitional form of analysis is employed--the system simply checks whether each student's action is a known step in a known plan, and tracks the student's progress through the plan. Actions that the system does not recognize are ignored, unless they have an undesirable effect on the state of one or more devices. The analysis becomes increasingly recognitional over time, because whenever the system employs the expert cognitive model to analyze the situation, it remembers the results of the analysis for use in similar situations.

We estimate that there are many real-world skills where feedback from the environment can guide the problem solving problem solving process as in REACT. Intelligent tutoring systems tend to overlook the role of the environment because they are frequently applied to abstract domains such as geometry or subtraction. Even in these domains there may be useful environmental cues to exploit. For example, intelligent tutors for programming tend not to take advantage of feedback from actually running the student's program, also recent work such as GIL is making such feedback more readily available to the student (Reiser et al., 1989)

Example Problem

To illustrate how REACT works we will now describe an example from our task domain. Students are assigned missions that involve activities such as configuring and calibrating a set of communications devices, establishing a link to a spacecraft, recording data from the spacecraft, and transferring the recorded data to a control center. These tasks involve sending commands asynchronously via a computer terminal over a local area network to the devices. Standard command sequences for each type of mission are defined by procedure manuals. The devices initially respond to each command with an indication of whether the command is accepted or rejected; if accepted, the devices require time to change state.

Figure 1 shows two procedures. The first procedure, Configure-DSP, is used to configure the DSP subsystem, which is used for spectrum processing. The steps mostly involve loading or setting parameters and selecting devices. The second procedure, Coherence-Test, is used to test the continuity and coherence of the communications link; it is supposed to be executed after the Configure-DSP procedure has been completed.

We will walk through the example shown in Figure 2 to illustrate how REACT overcomes the impediments to plan recognition. Here a student begins with Configure-DSP's first command for loading the predicts file, NLOAD JK. Line 1 shows the NLOAD command, and line 2 shows the device's response, COMPLETED, indicating that the command was accepted. Everything is proceeding as predicted by the plan: the correct command was issued by the student and it was accepted by the device.

Things get a bit more complicated on lines 3 through 7. On line 3 the student issues the next command in the Configure-DSP plan, NRMED. This command follows the Configure-DSP plan exactly, but the situation actually requires a different action to be taken, LD0 E, (i.e., enable recorder LD0), which is why the command is rejected on line 4. REACT thus must recognize when deviations from the plan are warranted; it does this by first noting the rejection and reasoning about why the action was not appropriate. In this case the command was rejected due to

Configure-DSP		Coherence-Test	
Command	Description	Command	Description
NLOAD *x*	load-predicts-file	NPCG *x*	set NPCG mode
NRMED *x*	select-recorder	NRUN *x*	run NCB program
SAT *x*	S-band attenuation	NDTE *x*	enable DTE
NTOP *x y*	set temperature	NFFT *x*	enable NFFT
OFST *x*	set offset time		

Figure 1: Example procedures

#	Commands / Responses	REACT's Explanation
1	> NLOAD JK	
2	> COMPLETED.	
3	> NRMED LD0	
4	> REJECTED. LD0 DISABLED	The NRMED command failed because one of its preconditions was unsatisfied: LD0 should be in the ONLINE mode instead of the OFFLINE mode. To resolve the impasse, Issue the command: LD0 E, Then Issue the command: NRMED LD0
5	> LD0 E	
6	> COMPLETED. LD0: ONLN	
7	> NRMED LD0	
8	> COMPLETED.	
9	> SAT 55	
10	> COMPLETED.	
11	> NTOP 20.0 30.0	
12	> COMPLETED.	
13	> NPCG MAN	
14	> COMPLETED.	You started the Coherence-Test procedure before you finished the Configure-DSP procedure. Issue the Command: OFST <>
15	> OFST 2.7	
16	> COMPLETED.	
17	> NRUN COLD	
18	> COMPLETED.	You failed to achieve one of the goals of the Configure-DSP procedure: SAT = 12. Issue the command: NIDLE REC, Then Issue the command: SAT 12
19	> NDTE E	
20	> COMPLETED.	
21	> NFFT E	
22	> COMPLETED.	

Figure 2: An example of tutoring with REACT

an action constraint violation (i.e., an unsatisfied precondition) by the NRMED command. REACT explains its reasoning about the violation as well as deriving a way to resolve the difficulty. The difficulty is viewed as an impasse because it prevents the student from continuing with the procedure, and it suggests a gap in the student's knowledge--if he had a good grasp of the procedure, he would have known to check the state of recorder LD0 before selecting it. At line 7 the student issues the NRMED command a second time; the plan calls for it to be issued just once. The second occurrence of the command is determined to be appropriate given that the first attempt at this action failed.

The example next illustrates difficulties that arise when the student follows a plan but fails to achieve its goals. The commands and responses on lines 9 through 14 follow the Configure-DSP plan exactly and all of the commands are accepted by the device. However, the parameter value of the SAT (Set S-Band attenuation value) command, 55, will not achieve one of the procedure's goals, that the value should be 12 by the time of the procedure's completion. This goal is not explicitly stated in the procedure, rather, it is derivable from the mission support data provided to the student. If the student does not correct this setting, it will affect the quality of the communications link with the spacecraft and of the data being recorded. Failure to achieve a goal is another type of impasse that can occur when a student is performing a task, indicating another type of knowledge gap in the student's skill set. REACT gives the student the opportunity to correct the error alone, but will intervene if not, before it is too late to correct it. When it detects the NRUN COLD (i.e., run NCB program) command on line 19, that belongs to the Coherence-Test plan, it initiates the interaction concerning the unsatisfied goal. In this case REACT also employs its expert cognitive model to analyze the cause of the impasse and determine a solution.

The final point made by the example centers on the actions listed on lines 15 through 18. On line 15 the student sends the NPCG MAN (i.e., set the NPCG device to manual mode) command, which is the first command in the Coherence-Test procedure, prior to finishing the Configure-DSP procedure, which has OFST (set the offset time) as its last command. This is a straightforward case of misordered plans, and REACT immediately alerts the student that a step was missed prior to starting the new procedure (see line 16). REACT recognizes this type of impasse as a plan dependency violation.

Situated Plan Attribution

Three types of impasses were introduced in the above example: (a) action constraint impasses, where the student takes an action that is in the plan but which the situation does not warrant, (b) goal failure impasses, where the student completes a plan without having achieved its goals, and (c) plan dependency impasses, where the student executes a plan before successfully completing one of its required predecessors. We will now give the details of how REACT recognizes and resolves each of these types of impasses.

Soar cognitive architecture

REACT is implemented in Soar, a problem solving architecture that implements a theory of human cognition (Laird et al., 1987; Newell, 1990). Soar is an integrated problem solving and learning architecture. Tasks in Soar are represented and performed in problem spaces. A Soar problem space consists of a collection of operators and states. Operators are proposed, selected, and applied to the current state; the resulting state changes may cause other operators in the problem space to be proposed, selected, and applied. Impasses occur in Soar when the problem solver stops making progress. To resolve an impasse, the Soar problem solver creates a subgoal and selects a different problem space where other operators are available for solving the problem. When the subgoal problem solving is successful, the results are saved in new productions created by Soar's chunking mechanism, which also saves the conditions that led to the impasse in the first place. The next time the conditions occur the learned chunk will be applied instead of having to search for an operator in the goal hierarchy.

Knowledge representation in REACT

REACT models several other aspects of plans besides the component actions shown in Figure 1, as will be briefly described below. For each type of mission the temporal precedence relationships among the plans is modeled with a directed graph structure called a temporal dependency network (TDN) (Fayyad&Cooper, 1992). A plan has a name and three attributes: state, execution status and goal status. The state of a plan can be either ACTIVE or INACTIVE; a plan is considered to be active once all of its predecessors in the TDN have been successfully completed. It is inactive prior to being active, and it becomes inactive again once it has been successfully completed. A plan's execution status (INCOMPLETE or COMPLETE) is determined by whether all of its commands have been observed. Each plan's goal status is marked satisfied if all its goals have been satisfied, otherwise it is unsatisfied.

Plans have two entities associated with them: operators (commands) and goals. The operators for the plans named Configure-DSP and Coherence-Test are shown in Figure 1. Each operator has a set of preconditions. A precondition is a tuple representing a device state that must be true before it can be considered satisfied. Similarly, a plan goal is also a tuple that represents a device state. As will be seen in the following sections, an active plan's goals are individually monitored for satisfaction at all times.

Problem solving organization of REACT

The problem solving in REACT is organized into two high-level activities: *plan tracking* and *impasse interpretation* . Plan tracking involves watching the student interact with the environment and deciding whether the student's actions are appropriate for the assigned task and situation. An inappropriate action is classified into an impasse category, and REACT interprets the impasse by executing an expert cognitive model to explain the impasse and suggest repairs to the procedure that will overcome it. The resulting explanation is used for tutoring the student. Plan tracking is implemented by performing the following activities:

- Perceive objects in the environment: REACT must continuously monitor the attributes of each of the objects in the environment. The perceive-object operator in Figure 3 initially perceives each of the objects in the environment and registers them in REACT's working memory. These attribute values are change as the objects are observed to change state in the environment.
- Monitor and evaluate student actions: Each of the student's actions is observed and matched with a plan. During the plan matching, preference is given to ACTIVE plans over INACTIVE plans. Likewise, the effects of the action on the device are also evaluated to determine whether the action was successfully completed or not. If an action was unsuccessful, it is immediately classified as an action-constraint impasse. Regardless of the action's outcome, the plan containing the action is marked with the match. If the action does not match any ACTIVE plan but does match with an INACTIVE plan, then REACT recognizes that a plan-dependency impasse has occurred. All the activities for evaluating individual actions are performed by the analyze-action-response operator and its associated problem space.
- Monitor individual goal status: The achievement status of the individual goals of ACTIVE plans is continually monitored. A plan's goals begin to be monitored when the plan becomes ACTIVE; monitoring ends when the plan in INACTIVE. The recognize-desired-results operator tests for satisfied goals, while the recognize-undesired-results operator tests for unsatisfied goals.
- Monitor conjunctive goal status: In addition to testing for the satisfaction of individual goals, REACT also continually monitors whether the conjunction of a plan's goals has been achieved. The recognize-goal-completion operator tests for the conjunctive satisfaction of a plan's goals.
- Monitor plan execution status: Besides monitoring whether a plan's goals have been achieved, REACT also

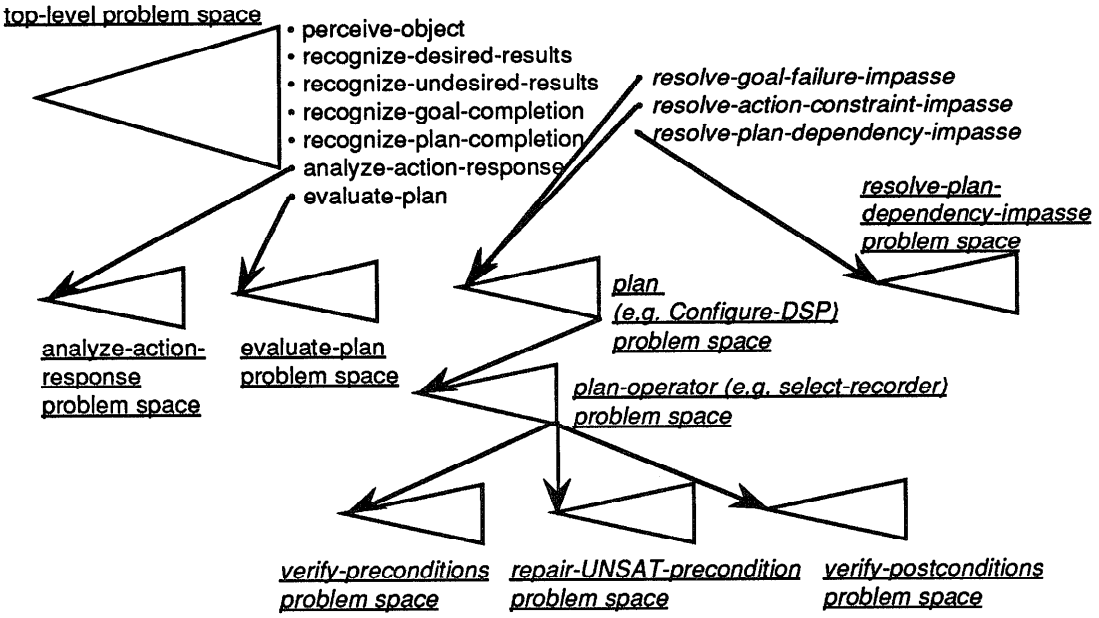

Figure 3: REACT's problem space hierarchy

monitors whether all of a plan's actions have been matched. The plan is marked COMPLETED once all of its actions have been observed. The recognize-plan-completion operator is responsible for monitoring a plan's execution status.

• Evaluate completed plans: If a plan marked COMPLETED has unsatisfied goals, then REACT recognizes that a goal-failure impasse has occurred. The evaluate-plan operator is used to perform this analysis.

Impasse interprctation, which is performed by the italicized Soar problem spaces shown in Figure 3, is implemented by performing the following activities:

• Resolve a plan dependency impasse: REACT first determines which plans should have been ACTIVE at the time that the student took the inappropriate action. The situation may have warranted a deviation from the standard procedure, e.g., the precondition of one of an action was unsatisfied. REACT checks the effects of the student's actions to determine whether it satisfies a precondition or else satisfies a goal of one of the plans. If either of these cases is true, the impasse is interpreted to be a situationally warranted deviation, and no further interpretation is necessary. On the other hand, if the action can not be justified by the situation, then REACT notifies the student of the violation of the plan ordering.

• Resolve a goal failure impasse: REACT selects the plan where the goal failure impasse occurred and determines how to achieve the unsatisfied goals. It reviews the plan, which contains some history of how the student executed its actions, by checking the operators related to the unachieved goals to determine whether the student sent the correct parameters with the action. REACT internally simulates the execution of the plan by selecting the appropriate action operators, verifying its preconditions are satisfied, repairing any unsatisfied preconditions, and verifying that the actions postconditions are satisfied. To repair an unsatisfied precondition may involve recursively selecting and applying other actions in the same manner. REACT follows this process, which is graphically portrayed in Figure 3, until the plan's goals are achieved. In the process of solving the problem, REACT generates an explanation for tutoring the student about the impasse.

• Resolve an action constraint impasse: REACT handles this type of impasse in much the same way as a goal failure impasse. The main difference is that it focuses on how to correctly apply a single operator within the plan rather than on the achievement of the goals of the whole plan.

The problem solving in REACT bears a resemblance to the approach in CHEF (Hammond, 1990): the general strategy is to notice a failure, build an explanation for it, use the explanation to determine a repair strategy, and so on. REACT's repair strategy emphasizes treating failures related to unsatisfied preconditions, which is similar to CHEF, but REACT generates repairs on the fly rather than using a case-based approach. In addition, certain types of repairs done by CHEF are not appropriate in REACT because they would imply intervening before the impasse was clear to the student. REACT permits the student to fix problems; the tutor only intervenes when it is clear that the student has reached an impasse in problem solving.

Example revisited

To illustrate how REACT works, we will revisit the example used in section 4, focusing on the action constraint impasse that occurred on lines 3 and 4, where

the student issued the NRMED LD0 command and it was subsequently rejected. The command-response pair on lines 3 and 4 is detected by the analyze-action-response operator, which subgoals into the analyze-action-response problem space where the NRMED command is matched to the active plan called Configure-DSP (Figure 1). Since the command was rejected by the simulator, the operator sets a flag indicating an action constraint impasse and the subgoal terminates. Then the resolve-action-constraint-impasse operator is selected and a subgoal into the Configure-DSP Plan problem space is formed. The operator corresponding to the NRMED command called select-recording-device is selected and another subgoal is made into the select-recording-device problem space (shown as Plan Operator Problem Space in Figure 3.) A subgoal into the Verify-Preconditions problem space is made for each of the select-recording-device operator's preconditions. As it turns out, the precondition that says that the recording device being selected must be in the ONLINE mode is unsatisfied. This is where the first part of the explanation on line 4 in Figure 2 is generated. Next, REACT subgoals into the Repair-UNSAT-Precondition problem space, where it is determined that issuing LD0 E (enable recording device LD0) command will satisfy the precondition. This information is also put into the explanation on line 4 of the example. Finally, once the precondition is satisfied, the select-recording-device problem space simulates sending the NRMED LD0 command (select the recording device named LD0), and this is also added to the explanation and this subgoal terminates. Since REACT has determined how to resolve the impasse, all of the subgoals in the hierarchy terminate and the impasse recognition operators resume their work with the next action-response pair.

Evaluation

A pilot study was conducted to evaluate REACT. We hypothesized that situated plan attribution would be able to recognize and interpret student impasses, and that the resulting tutoring would help students acquire skill in the LMC domain. The study was conducted with seven students of approximately equal experience who were divided into two groups. Each group of students was assigned a task to perform on the LMC simulator. The difference between the two groups was that Group I's students were tutored by REACT during the exercise, while Group II's students did not receive any tutoring. To test the hypotheses about situated plan attribution and student learning at impasse points, the tasks were configured so that certain types of action constraint and goal failure impasses would occur if the student was not experienced, which was the case with both groups.

The results of the study suggest that situated plan attribution holds promise as a method for recognizing student impasses and for explaining how to resolve them in a satisfactory manner. During the study REACT interpreted 604 different command-response pairs (actions) performed by the students. It recognized and explicated 36 action constraint impasses, 5 plan dependency impasses, and 17 goal failure impasses. In analyzing the event logs, REACT did not make any misinterpretations, and it was able to make all of its analyses quickly enough for a timely interaction with the student.

The study also suggests that the way situated plan attribution was applied (i.e., for impasse-driven tutoring) helped students to improve their skills in the LMC domain more quickly than students without tutoring. The students from both groups reached impasses, but there was a significant difference between the two groups in the amount of time it took resolve these impasses. While both groups acquired the same amount of skill in cases where there was an action constraint violation, the students in Group I (with REACT) resolved impasses and acquired the new knowledge approximately ten times faster than the students in Group II. Likewise, the students in Group I were less prone to having certain goal failures than the students in Group II. It was observed that students who did not notice a goal failure the first time they performed a task were prone to never realizing that there was one. For more details on the pilot study, see (Hill, 1993).

Finally, though REACT was shown to be robust in the task domain we have described in this paper, we suspect that it will be necessary to make some improvements to the situated plan attribution problem spaces to cope with larger numbers of plans and actions. In these cases we anticipate the need to deal with more ambiguity than was present in our current implementation. Ambiguity primarily will have an impact on the interpretation of plan dependency violation impasses---REACT might have to delay offering assistance until it is clear which plan the student is attempting next.

Conclusions

We have introduced a plan recognition technique called situated plan attribution that we claim avoids some of the problems of other approaches, especially as applied to intelligent tutoring. Specifically, we have shown how our method is flexible enough to recognize when a situation warrants an action that is not specified by a plan. Likewise, it recognizes when an action specified by a plan is not situationally appropriate.

Situated plan attribution also addresses the issues of underconstrained and unfocused modeling in that it concentrates on recognizing students' impasse points rather than trying to generate or understand the mental states that led to a particular action. Impasse points are natural places to tutor, and the amount of processing required to recognize and explicate the impasses we have defined is reasonable.

Acknowledgments

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Dr. Johnson was supported in part by the Advanced Research Projects Agency and the Naval Research Laboratory under contract number N00014-92-K-2015 (via a subcontract from the University of Michigan). Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of the U.S. Government or any agency thereof.

References

(Anderson et al., 1990) John R. Anderson, C. Franklin Boyle, Albert T. Corbett and Matthew W. Lewis. Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42, 1990: 7-49.

(Azarewicz et al., 1986) J. Azarewicz and Fala, G. and Fink, R. and Heighecker, C.., Plan Recognition for Airborne Tactical Decision Making, *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 805-811, 1986.

(Brown&VanLehn, 1982) John Seely Brown and Kurt VanLehn. Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 4, 1982:379-426.

(Calistri, 1990) Randall J. Calistri. Classifying and detecting plan-based misconceptions for robust plan recognition. Ph.D. diss., Technical Report No. CS-90-11, Department of Computer Science, Brown University, 1990.

(Corbett et al., 1990) Albert T. Corbett, John R. Anderson, and Eric G. Patterson. Student Modeling and Tutoring Flexibility in the Lisp Intelligent Tutoring System. *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education.* Ablex, 1990.

(Charniak&Goldman, 1991) Eugene Charniak and Robert Goldman. A Probabilistic Model of Plan Recognition. *Proceedings of the Ninth National Conference on Artificial Intelligence.* 1991.

(Fayyad&Cooper, 1992) Kristina Fayyad and Lynne Cooper. Representing operations procedures using temporal dependency networks. *Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations*, SPACEOPS-92, Pasadena, CA, November 16-20, 1992.

(Hammond, 1990) Kristian J. Hammond. Explaining and Repairing Plans That Fail*. *Artificial Intelligence*, 45, (1990) 173-228.

(Hill, 1993) Randall W. Hill, Jr.. "Impasse-driven tutoring for reactive skill acquisition." Ph.D. diss. University of Southern California, Los Angeles, California, 1993.

(Hill&Johnson, 1993a) Randall W. Hill, Jr. and W. Lewis Johnson. Designing an intelligent tutoring system based on a reactive model of skill acquisition. *Proceedings of the World Conference on Artificial Intelligence in Education (AI-ED 93)*, Edinburgh, Scotland, 1993.

(Hill&Johnson, 1993b) Randall W. Hill, Jr. and W. Lewis Johnson. Impasse-driven tutoring for reactive skill acquisition. *Proceedings of the 1993 Conference on Intelligent Computer-Aided Training and Virtual Environment Technology (ICAT-VET-93)*, NASA/Johnson Space Center, Houston, Texas, May 5-7, 1993.

(Johnson, 1986) W. Lewis Johnson. *Intention-based diagnosis of novice programming errors.* Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

(Kautz&Allen, 1986) Henry A. Kautz and James F. Allen. Generalized plan recognition. *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

(Laird et al., 1987) John E. Laird, Allen Newell and Paul S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33, 1987:1-64.

(Newell, 1990) Allen Newell. *Unified Theories of Cognition.* Harvard University Press, 1990.

(Reiser et al., 1989) B.J. Reiser, M Ranner, M.C. Lovett, and D.Y. Kimberg. Facilitating Students' Reasoning with Causal Explanations and Visual Representations. *Artificial Intelligence and Education: Proceedings of the 4th International Conference on AI and Education.*, pp. 228-235. Amsterdam: IOS, 1989.

(Self, 1990) John A. Self, "Bypassing the Intractable Problem of Student Modeling", in Frasson, C. and Gauthier, G., eds., *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, Ablex, Norwood, NJ, pp. 107-123, 1990

(VanLehn, 1982) Kurt VanLehn. Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. *The Journal of Mathematical Behavior*, 3, 1982:3-71.

(VanLehn, 1983) Kurt VanLehn. *Felicity conditions for human skill acquisition: Validating an AI-based theory.* Tech. Report CIS-21. Palo Alto, CA: Xerox Palo Alto Research Center.

(VanLehn, 1988) Kurt VanLehn. Toward a theory of impasse-driven learning. *Learning Issues for Intelligent Tutoring Systems.* Edited by Heinz Mandl and Alan Lesgold. New York: Springer-Verlag, 1988:19-41.

(Ward, 1991) Blake Ward. ET-Soar: Toward an ITS for theory-based representations. Ph.D. diss., CMU-CS-91-146, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.