Situated Plan Attribution for Intelligent Tutoring

Randall W. Hill, Jr.
CalTech / Jet Propulsion Laboratory
4800 Oak Grove Drive M/S 525 3960
Pasadena, CA 91109-8099
hill@gobi.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 and agent modeling techniques are potentially useful for intelligent tutoring, but are difficult to employ in practice. However, 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. Situated plan attribution interprets both the student's actions and the environment in which the student is situated. 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. Cognitive modeling studies have indicated that tutorial interaction maximally effective at impasse points. This approach has been implemented and evaluated in the context of the REACT tutor, a trainer for operators of deep space communications stations.

Topic areas: computer-aided education, plan recognition, cognitive modeling

Acknowledgements

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.

Situated Plan Attribution for Intelligent Tutoring

Abstract

Plan recognition and agent modeling techniques are potentially useful for intelligent tutoring, but are difficult to employ in practice. However, 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. Situated plan attribution interprets both the student's actions and the environment in which the student is situated. It devotes varying amounts of effort to the interpretation process, focusing the greatest effort on interpreting *impasse points*, i.e., points where the the student encounters some difficulty completing the task. Cognitive modeling studies have indicated that tutorial interaction maximally effective at impasse points. This approach has been implemented and evaluated in the context of the REACT tutor, a trainer for Operators of deep space communications stations.

Topic areas: computer-aided education, plan recognition, cognitive modeling

1. Introduction

Plan recognition and agent modeling capabilities are valuable for intelligent tutoring (Corbet 1 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., 1 986). 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 *unfocussed*---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.

2. 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.

3. 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)

4. 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.

Configure-DSP		Coherence-Test	
Command	Description	Command	Description
NLOAD *x1*	load-predicts-file	NPCG *x1*	set NPCG mode
NRMED *x1*	select-recorder	NRUN *x1*	run NCB program
SA1 *x1*	S-band attenuation	NDTE *x1*	enable DTE
XA1 *x1*	X-band attenuation	NFF1 *x1*	enable NFF1
NFOP *x1x2*	set temperature		
OFS-I *x1*	set off set time		

Figure 1:1 Example Procedures

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.

#	'Commands / Responses	REACT's Explanation
1	> N1OA[)JK	
2	>COMPLETE D.	
3	> NRMED 1 D0	
4	>REJECTED. 1 DO DISABLED	'1 he NRMED command failed because one, of its preconditions was unsatisfied: LD0 should be in the ONLINE mode instead of the OFFLINE mode. 10 resolve the impasse, issue the command: LD0E then Issue the command: NRMED 1 D0
5	> LD0 E	
6	> COMPLETED. LD0: ONLN	
7	> NRMED LD0	
8	> COMPLE'1 E D.	
9	> SA1 55	
10	> COMPLE'1 E D,	
11	>XA1 13	
12	> COMPLE 1 E D.	
13	> NTOP 20,030,0	
14	> COMPLE 1 E D.	
15	> NPCG MAN	
16	> COMPLETE D,	You started the Coherence-lest procedure before you finished the Configure-DSP procedure. Issue the Command: OF S1 <>
1-/	> OF ST2,"7	
18	> COMPLE 1 E D.	
19	> NRUNCOL D	
?0	> COMPLE "1 E D.	You failed to achieve one of the goals of the Configure-DSP procedure: SA1 = 12, Issue the command: NIDLE RE C .hen Issue the command: SA1 12
21	>NDTE E	
2?	>COMPLE'1 E D,	
23	> NF F T E	
24	> COMPLE "1 E D.	

Figure 2: An example of tutoring with REACT

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 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 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 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.

.5. Situated Plan Attribution

Three types of impasses were introduced in the above example: (a) action constraint impasse, where the student takes an action that is in the plan but

which the situation does not warrant, (b) goal failure impasse, where the student completes a plan without having achieved its goals, and (c) plan dependency impasse, 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.

5.1 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.

5.2 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 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.

5.3 Problem space description of REACT

The problem space organization of REACT is depicted in Figure 3. REACT is an integrated problem solving system, but for the purposes of this discussion the operators and problem spaces can be functionally divided into two categories: impasse recognition and impasse explication.

The operators in the top-level problem space that are responsible for recognizing impasses are shown with bullets and non-italicized names in Figure 3. The first of these is the perceive-object operator. This operator is used to add external objects, in this case the devices and their attributes, to REACT's internal model of the world. Once a device has been added, any changes to the device's state are automatically updated in REACT's internal model.

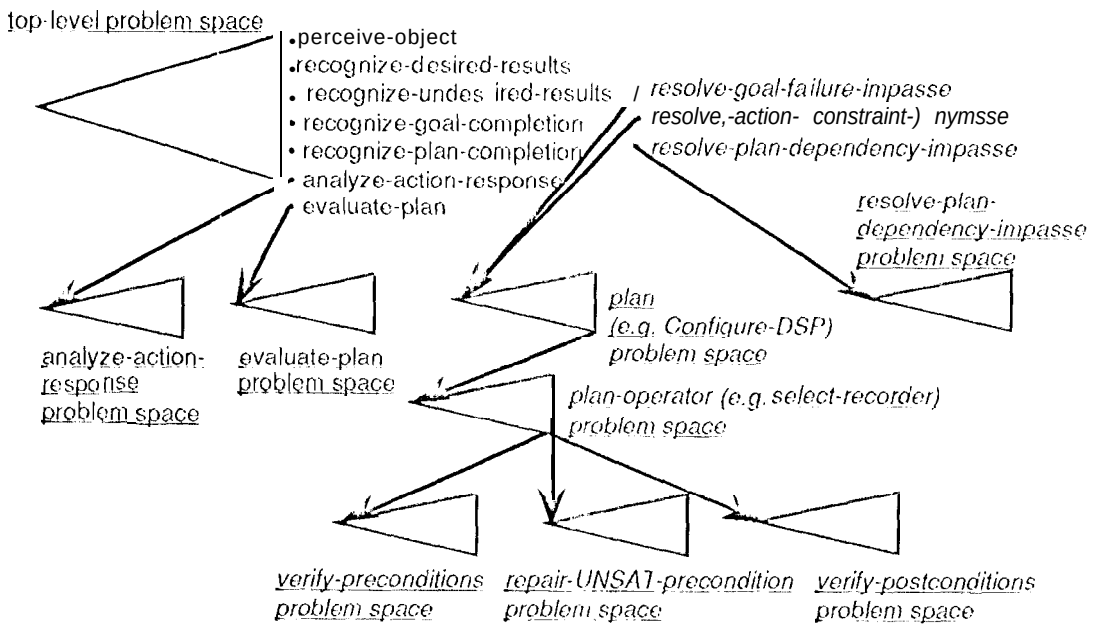

Figure 3: REACT "s 1 'roblem Space 1 lierarchy

The next four operators, recognize-desired-results, recognize-undesired-results, recognize-goal-completion, and recognize-plan-completion are used to continually monitor the status of active plans and their goals. (Note: more than one plan may be active at a time.) The recognize-desired-results and

recognize-undesired-results operators are used for keeping track of the individual goals of an active plan. These operators are selected and applied as soon as a relevant device state changes. The recognize-goal-completion operator is selected and applied when the conjunction of all of an active plan's goals is satisfied. Likewise, the recognize-plan-completion operator changes a plan's status to COMPLETE when all of the plan's operators have been observed. The operator called analyze-action-response is selected and applied each time the student takes an action. The analyze-action-response operator subgoals to a problem space with the same name, where there is a set of operators that match the student's command to a plan. When matching the command to a plan, preference is given to active plans over inactive ones, since we expect the student to be performing an active plan. If the command was accepted by the device, then the plan's corresponding operator is marked and the subgoal is terminated.

If the student's command was rejected, then the plan's operator is marked, but a flag is also raised to indicate that an action-constraint impasse has occurred. Likewise, if a command does not match with any active plan, but does match an inactive plan, then a flag is raised to indicate that the student is at a plan dependency impasse. In either case, once the matching has been completed, the analyze-action-response subgoal terminates and problem solving continues in the top-level problem space.

Before discussing how REACT handles an impasse once it has been discovered, we will cover the last major operator used for impasse recognition, evaluate-plan. This operator is selected and applied after an active plan's status has been changed to COMPLETE, i.e., all of the plan's operators have been matched with student actions. A subgoal is formed and the evaluate-plan problem space is selected where there are operators that check the plan's goal status. If the goal status is SATISFIED, then the plan is made inactive, and operators are applied to activate other eligible successors to the plan being evaluated. If the plan's goal status is UNSATISFIED then an operator raises the flag for a goal failure impasse. The subgoal then terminates and problem solving continues in the top-level problem space.

Now we turn to the other major problem solving activity that can be initiated from the top-level problem space, namely, impasse explication. There is one operator for each type of impasse, shown in italics in Figure 3 In the case of a plan dependency impasse the explication process is simple: a subgoal is formed by the resolve-plan-dependency-impasse operator, where an explanation is generated for the student telling about the unfinished plan, including which commands have yet to be sent.

For the other two impasse categories, action-constraint and goal failure, the expert cognitive model is invoked at the point in the problem solving

where the impasse was detected. (The problem spaces implementing the expert cognitive model are shown in italics.) The expert cognitive model puts itself in the student's situation and simulates either taking an individual action or taking whatever actions are necessary to achieve a plan's goals, depending on the type of impasse. The problem solving in either case involves selecting the plan (Plan Problem Space) where the impasse occurred, selecting an operator from that plan (Plan-Operator Problem Space), and then verifying the operator's preconditions (Verify- Preconditions Problem Space) with respect to the device state. If one or more preconditions is not satisfied the cognitive model subgoals into the Repair-UNSAT-precondition problem space where it determines what actions to take to satisfy them. In the process, explanation is generated.

5.4 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.

5.5 Role of learning

REACT improves its performance as it gains experience recognizing impasses in different situations. Recall that Soar builds new productions each time that REACT subgoals into one of its problem spaces for impasse recognition or impasse explication. The new productions summarize the conditions that led to the subgoal in the first place as well as the end results of the search. REACT therefore avoids subgoaling in situations where it has previously recognized and explicated an impasse. Instead when a command-response pair is observed REACT immediately applies a recognition chunk that determines whether there is an impasse or not instead of subgoaling into the analyze-action-response problem space. Once this chunk fires, there are explication chunks that generate an explanation for the interaction with the student without activating the search through the expert cognitive model problem spaces. As chunks are built the lines between impasse recognition and impasse explication begin to blur since much of the problem solving is done in the top-level problem space and not in the different branches of the problem space hierarchy. This is somewhat different from the traditional intelligent tutoring model that makes a distinct functional separation among the student, expert and tutoring models (Warren&Goodman, 1993).

6. Evaluation

A pilot study was conducted to evaluate REACT, both in terms of its ability to recognize and explicate impasses and its ability to abet students' learning, The study had seven students who were divided into two groups. Group 1's students were tutored by REACT while they performed assigned tasks on a simulator, and Group 11's students performed the identical tasks without REACT. Pre-tests indicated that the two groups had roughly the same skill level at the beginning of the study. 'The tasks were configured so that certain types of action constraint and goal failure impasses would occur if the student was not eqx'rimed, which was the case with both groups. 'The results of the evaluation indicated a significant difference in the amount of time it took to acquire skill at impasse points. While both groups acquired the same amount of skill in cam where there was an action constraint violation, the students in Group I (with REACT) resolved the 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 making

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.

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.

A questionnaire was also administered to the test subjects, who generally found REACT to be helpful and understandable, but who also found some of the explanations to be cryptic. We have decided it would have been helpful to explain the impasse categories prior to the training as well as the representation that was used for device state.

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 that 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.

7. 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 anti explicate the impasses we have defined is reasonable,

8. 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 & 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.

(Corbet 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.

(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 ofthe 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.

(Warren&Goodman, 1993) Kimberly C. Warren and Bradley A. Goodman. Engineering intelligent tutoring systems. *1993 Conference on Intelligent Computer-Aided Training and Virtual Environment Technology, (ICAT-VET-93)*, NASA Johnson Space Center, Houston, Texas, May 5-7, 1993.