Procedural help in Andes: Generating hints using a Bayesian network student model

Abigail S. Gertner and **Cristina Conati** and **Kurt VanLehn**
Learning Research & Development Center
University of Pittsburgh
gertner+@pitt.edu, conati@pogo.isp.pitt.edu, vanlehn+@pitt.edu

Abstract

One of the most important problems for an intelligent tutoring system is deciding how to respond when a student asks for help. Responding cooperatively requires an understanding of both what solution path the student is pursuing, and the student's current level of domain knowledge. Andes, an intelligent tutoring system for Newtonian physics, refers to a probabilistic student model to make decisions about responding to help requests. Andes' student model uses a Bayesian network that computes a probabilistic assessment of three kinds of information: (1) the student's general knowledge about physics, (2) the student's specific knowledge about the current problem, and (3) the abstract plans that the student may be pursuing to solve the problem. Using this model, Andes provides feedback and hints tailored to the student's knowledge and goals.

Introduction

Many different kinds of computer programs have to decide how to respond when their users ask for help, and some must even decide when help is needed. Both of these tasks involve a great deal of uncertainty, especially in the case of Intelligent Tutoring Systems (ITS), where there is uncertainty about both the student's intentions and what the student knows about the task domain.

The problem we address in this paper is how to decide what to say when a student needs help solving a problem, given observations of what the student has done already. Our solution uses a probabilistic model of the student's knowledge and goals to decide between alternatives. We have developed a procedure that searches the solution space of the problem the student is working on to find a proposition that is both part of the solution path the student is probably pursuing, and that the student is unlikely to know. This proposition will be the subject of the help given to the student. Furthermore, we use a theory of hinting to model the effect of the help that has been given on the student's mental state. This framework for responding to help requests is implemented in Andes, an ITS for Newtonian physics.

Andes' tutor uses *coached problem solving* (VanLehn 1996), a method of teaching cognitive skills in which the tutor and the student collaborate to solve problems. In coached problem solving, the initiative in the student-tutor interaction changes according to the progress being made. As long as the student proceeds along a correct solution, the tutor merely indicates agreement with each step. When the student stumbles on part of the problem, the tutor helps the student overcome the impasse by providing hints that lead the student back to a correct solution path. In this setting, a critical problem for the tutor is to interpret the student's actions and the line of reasoning that the student is following so that it can conform its hints to that line of reasoning.

This paper first describes how Andes' probabilistic student model is created and how it represents various aspects of the student's mental state while solving a problem. We then demonstrate how Andes uses this student model to generate hints that are both relevant and appropriate to the student's understanding of the domain.

The Andes Tutoring System

Andes has a modular architecture, as shown in Figure 1. The left side of Figure 1 shows the authoring environment. Prior to run time, a problem author creates both the graphical description of the problem, and the corresponding coded problem definition. Andes' problem solver uses this definition to automatically generate a model of the problem solution space called the *solution graph*.

The right side of the figure shows the run-time student environment. The student interface, known as the Workbench, sends student entries to the Action Interpreter, which looks them up in the solution graph and provides immediate feedback as to whether the entries are correct or incorrect. More detailed feedback is provided by Andes' Help System. Both the Action Interpreter and the Help System refer to the student model to make decisions about what kind of feedback and help to give the student. The most important part of the student model is a Bayesian network (Pearl 1988) that is constructed and updated by the Assessor, and provides probabilistic estimates of the student's goals, beliefs, and knowledge (Conati *et al.* 1997). The student model also contains information about what problems the student has worked on, what interface features they have

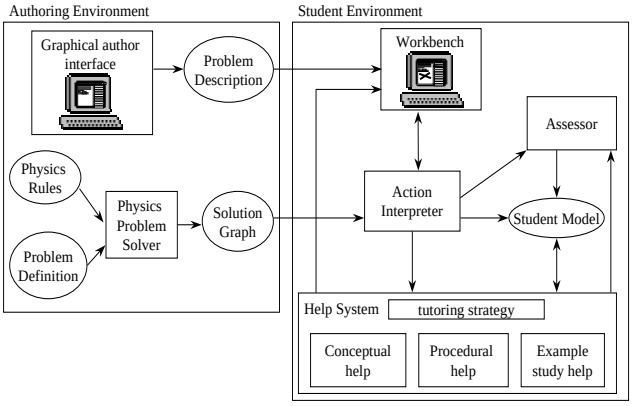


Figure 1: The Andes System Architecture. Rectangles are system modules, ellipses are data structures.

used, and what help they have received from the system in the past.

The Andes Student Modeling Framework

Inferring an agent's plan from a partial sequence of observable actions is a task that involves inherent uncertainty, since often the same observable actions can belong to different plans. In coached problem solving, two additional sources of uncertainty increase the difficulty of the plan recognition task. First, coached problem solving often involves interactions in which most of the important reasoning is hidden from the coach's view. Second, there is additional uncertainty regarding what domain knowledge the student has and can bring to bear in solving problems. While substantial research has been devoted to using probabilistic reasoning frameworks to deal with the inherent uncertainty of plan recognition (Charniak & Goldman 1993; Huber, Durfee, & Wellman 1994; Pynadath & Wellman 1995), none of it encompasses applications where much uncertainty concerns the user's planning and domain knowledge. On the other hand, probabilistic approaches to student modeling mostly assume certainty in plan recognition and use probabilistic techniques to model uncertainty about knowledge (Anderson *et al.* 1995; Jameson 1995).

Andes uses a framework for student modeling that performs plan recognition while taking into account both the uncertainty about the student's plans and the uncertainty about the student's knowledge state (Conati *et al.* 1997). By integrating these two kinds of information, Andes' student model is able to perform three functions: *plan recognition*, *prediction* of the student's future goals and actions, and *long-term assessment* of the student's domain knowledge. The framework uses a Bayesian network to represent and update the student model on-line, *during* problem solving (see Conati *et al.*, 1997, for a discussion of the issues involved in using Bayesian networks for on-line student modeling). In the following two sections we describe the structure of the student model and how it is created.

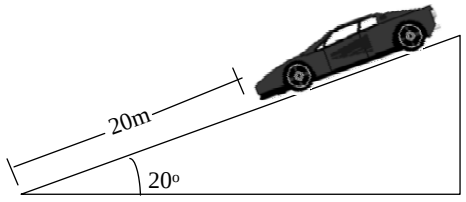
Generating the solution graph

Like its two predecessors, OLAE (Martin & VanLehn 1995) and POLA (Conati & VanLehn 1996), Andes automatically constructs its Bayesian networks from the output of a problem solver that generates all the acceptable solutions to a problem. We have based Andes' problem solver's rules on the representation used by Cascade (VanLehn, Jones, & Chi 1992), a cognitive model of knowledge acquisition developed from an analysis of protocols of students studying worked example problems. The rules are being developed in collaboration with three physics professors who are the domain experts for the Andes project.

In addition to knowledge about the qualitative and quantitative physics rules necessary to solve complex physics problems, Andes' problem solver has explicit knowledge about the abstract plans that an expert might use to solve problems, and about which Andes will tutor students. Thus, given an initial description of the problem situation and a problem-solving goal, Andes produces a hierarchical dependency network including, in addition to all acceptable solutions to the problem in terms of qualitative propositions and equations, the abstract plans for generating those solutions. This network, called the *solution graph*, represents Andes' model of the solution space.

For example, consider the problem statement shown in Figure 2. The problem solver starts with the top-level goal of finding the final velocity of the car. From this goal, it forms the sub-goal of using a kinematics equation, which involves several quantities including the car's acceleration and displacement. Since the acceleration of the car is unknown, the problem solver forms a sub-goal to find it, which in turn leads to a goal of using Newton's second law applied to the car.

When all applicable rules have fired, the result is a partially ordered network of propositions leading from the top-level goal to a set of equations that are sufficient to solve for the sought quantity. This network, including all propositions and the rules that were used to generate them, is saved as the solution graph. Figure 3 shows a section of the solution graph for this problem, showing the relationship between the goals of finding the final velocity and finding the acceleration, and the actions that address those goals.



A 2000kg car at the top of a 20° inclined driveway 20m long slips its parking brake and rolls down. Assume that the driveway is frictionless. At what speed will it hit the garage door?

Figure 2: A physics problem.

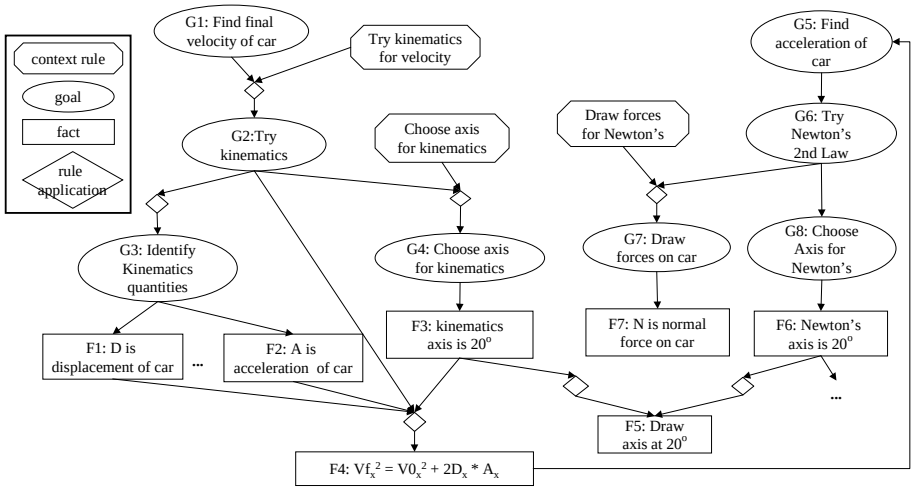


Figure 3: A solution graph segment for the car problem. Some rule nodes have been left out for readability.

The Assessor's Bayesian Network

The Assessor's Bayesian network is automatically generated each time the student selects a new problem. The structure of the Bayesian network is taken directly from the structure of the solution graph. The network contains five kinds of nodes, shown in Figure 3 using different shapes:

1. *Context-Rule nodes* model the ability to apply a rule in a specific problem solving context in which it may be used.
2. *Fact nodes* represent the probability that the student knows a fact that is part of the problem solution.
3. *Goal nodes* represent the probability that the student has been pursuing a goal that is part of the problem solution.
4. *Rule-Application nodes* represent the probability that the student has applied a piece of physics knowledge represented by a context-rule to derive a new fact or goal.
5. *Strategy nodes* (not shown in Figure 3) correspond to points where the student can choose among alternative plans to solve a problem.

To convert the solution graph into a Bayesian network, it is first annotated with prior probabilities for all the top level nodes – the rule nodes and propositions that were given in the problem statement. All other nodes are given a conditional probability table describing the relationship between the node and its parents. For example, each rule-application node has as parents exactly one rule node, corresponding to the rule that is being applied, and one or more goal and/or fact nodes, corresponding to the propositions that must be true in order to apply the rule. The conditional probability table of a rule-application node captures the assumption that the student will likely do the consequent action if all of the antecedent knowledge (Rule, Goals, and Facts) is available, but if any of the knowledge is not available, the student cannot apply the rule.

When a student performs an action in the Andes Workbench, the Action Interpreter determines which fact or goal nodes in the solution graph, if any, correspond to that action. If one or more are identified, the Assessor is told to set the value of those nodes to True, and the entire network is then re-evaluated to reflect the effects of the new observation.

In general, evidence that a fact is known causes the probabilities of the antecedents of the corresponding node(s) in the solution graph to go up, indicating the model's *explanation* for the new evidence. Likewise, the probabilities of goals and facts that are consequences of knowing the fact will also go up, corresponding to the model's *prediction* of future actions that have become more likely as a result of observing the evidence. If the student asks for help, Andes can use the probabilities produced by the Assessor to inform its decisions regarding the part of the solution graph about which to begin hinting.

Procedural help: deciding what to say

In a Wizard of Oz experiment designed to provide information about the kinds of help students using Andes might need (VanLehn 1996), students solved problems on an interface similar to Andes, requesting help by sending a message to a human tutor. In this experiment, the most common help request students made was of the form, "I'm stuck. What should I do next?" (27 occurrences out of 73 help requests). The part of Andes' help system that answers this kind of help request is called the Procedural Helper.

In most tutoring systems, such as Anderson's model tracing tutors (Anderson *et al.* 1995), it is easy to decide what the topic of a hint should be because the student is only allowed to follow one solution path. If there are several correct paths through the solution space, the tutor asks the student which one the student wants to pursue. Thus, the tutor always knows what path the student is on, and when

she indicates that she is stuck, the only possible hint is to point to the next performable step on that path.

In the physics task domain, however, there are many correct solution paths because inferences can be done in many different orders. In some problems, there may be more than one alternative solution strategy that may be brought to bear, resulting in the application of different physics laws and the use of different equations. Thus, it is impractical to keep asking the student which path she is following. In fact, our own informal analyses of tutoring transcripts indicate that human tutors rarely ask students what their goals are before giving a hint. Moreover, Andes seldom forces students explicitly to enter all the steps of a derivation in the interface. These properties of the domain make it very difficult to know what path the student is pursuing, and where along that path the student was when she got stuck.

Nonetheless, it would be extremely infelicitous if Andes gave hints intended to help a student along one part of the solution path when the student is actually focusing on a different part of the path, or possibly going down a different path altogether. Thus, Andes uses its Bayesian network to infer which part of the solution the student is working on and where she got stuck. This is a form of probabilistic plan recognition, and it is one of the main reasons for using Bayesian networks in Andes.

As with any plan recognition task, Andes needs an inductive bias to guide the search through the solution space for the student's most likely solution path. The bias that Andes uses is to assume that the student traverses the solution graph in depth-first order when generating a solution. This means that if a student has just identified the displacement of the car in Figure 3 (node F1 in the diagram), they would not be expected to go on to draw an axis (node F5) until they had also identified the car's acceleration (node F2). Following this assumption, Andes searches the solution graph depth-first for paths that begin with the student's most recent action.

Since we are trying to identify an appropriate part of the solution to give a hint about, we want to determine where the student is probably getting stuck. In other words, we have to find a node that the student is not likely to have in mind already. The depth-first traversal of the solution graph therefore will terminate whenever it reaches a node whose probability is below a certain threshold (currently 0.8). The search will also terminate if it reaches a node that must be entered before continuing with the rest of the solution because it is a precondition for applying any other rule along that path.

The result of this traversal is a set of paths through the solution graph, each beginning with the student's most recent action, and terminating with a node that has a probability of less than .8, or that must be entered. In our example (Figure 3), suppose that the last action observed is F5. Additionally, suppose the probabilities of F3, F6, G2, G3, G4, G6, and G8 are above .8, and the probabilities of F1, F2, F7, and G7 are below .8. The set of paths found in the part of the graph that is shown will be:

1. F5 → F3 → G4 → G2 → G3 → F1
2. F5 → F3 → G4 → G2 → G3 → F2
3. F5 → F6 → G8 → G6 → G7

Next, Andes must choose one of these paths as the one it will use to guide the student. To do this, it looks at the joint probability of all the nodes in each path *except* the last node, which is the one that the student is supposed to be stuck on. The path with the highest value is chosen as being the most likely to represent the student's current path, and thus the best candidate for procedural help.

In the absence of additional evidence, if the rules associated with kinematics have higher prior probability than the rules associated with Newton's law, then paths 1 and 2 will be chosen over the third path. However, since these two paths are identical except for the last node, they will have exactly the same joint probability. In such a situation, we need a metric to decide which of the last nodes in each path is the best candidate for a hint, given that both are on paths that the student is probably pursuing. We choose the node with the lowest probability, because it is the one that the student is most likely to be stuck on.

If, on the other hand, the student has performed some actions associated with the Newton's law plan, such as drawing a vector for the weight of the car (not shown in Figure 3), the third path will be more likely, and node G7 will therefore be selected as the topic of the hint to be generated.

Generating hints from BN nodes

Evidence from studies of the performance of human tutors suggests that one of the main reasons human tutors are effective is that they are able to let students do most of the work in correcting errors and overcoming impasses, while providing just enough guidance to keep them on a productive solution path (Merril *et al.* 1992). Likewise, in generating help from the target node selected by the procedure described above, Andes tries to encourage the student to solve the problem on her own by giving hints, rather than by directly telling her what actions to perform.

Andes' Procedural Helper uses templates to generate hints from nodes in the solution graph. For each goal and fact in its knowledge base, Andes has an associated *sequence* of hint templates, ranging from quite general to very specific. Slots in the templates are filled in with descriptions of the appropriate objects in the problem situation. When guiding a student towards a particular goal, Andes begins by using the most general templates to encourage the student to generate the next solution step with as little extra information as possible.

For example, suppose that Andes has selected node G7 from Figure 3, representing the goal of drawing all of the forces on the car, as the topic of its next hint as described in the previous section. The templates associated with this goal are:[1]

(3 "*Think about what you need to do in order to have a complete free body diagram for [ ].*" body)

[1]The numbers before each template indicate the specificity of the hint. The arguments after each template string tell the system how to fill in the corresponding slots, indicated by square brackets.

(5 *"Draw all the forces acting on [ ] as part of your free body diagram."* body)

Choosing the first template from this list, and substituting the appropriate descriptions of objects or quantities from the problem into the slots, Andes would generate the hint,

Hint 1 *"Think about what you need to do in order to have a complete free body diagram for [the car]."*

If the student does not know what to do after receiving the first general hint, she can select a follow-up question by clicking one of three buttons:

- Explain Further: Andes will display the next hint in the hint sequence, which gives slightly more specific information about the proposition represented by the node.
- How do I do that?: Andes finds a child of the hint node that has not yet been addressed, and gives a hint about that node. If there is more than one child node, it chooses the one with the lowest probability, assuming that is the node the student is most likely to be stuck on.
- Why?: Andes displays a canned description of the rule that was used by the problem solver to derive that node.

In the above example, after seeing Hint 1, clicking on the "Explain Further" button results in the hint,

Hint 2 *"Draw all the forces acting on [the car] as part of your free body diagram."*

which is a more specific description of the goal in question. Clicking "How do I do that?" after Hint 1, on the other hand, might result in the hint,

Hint 3 *"Do you know of any [other] forces acting on [the car]?"*

which points to a sub-goal of drawing the forces on the car, namely drawing the normal force (the word "other" is used optionally if at least one force has already been drawn).

The discourse model

Another important consideration when generating hints is the current discourse context. In particular, Andes should avoid giving a hint that the student has already seen. Therefore for each node in the solution graph, Andes keeps a record of what hints about that node it has given the student. When the hint selection algorithm selects a node that has already been mentioned, the Procedural Helper tries to give a more specific hint than was given last time. If the most specific hint available has already been given, Andes will repeat it.

Andes also uses its representation of what the student has done so far to generate referring expressions. For example, if the student has defined the acceleration of the car as A, Andes will refer to it by the variable A, rather than with its description. So Hint 1 above would be, "To find A, try using a principle that mentions acceleration."

Updating the student model after a hint

An ITS must take into account the hints that it has given when interpreting the student's actions and updating the student model. Typically, a student will ask for hints down to a certain level of specificity before taking the action suggested by the hint. Thus, the student modeler should interpret student actions taken in response to a hint differently depending on that hint's specificity.

This problem has been solved differently in different ITS's. Many tutors, e.g. (Anderson *et al.* 1995), assume that hints affect the knowledge directly. For instance, strong hints may cause the student to learn the knowledge required to make the action. Thus, it does not matter whether a student's correct entry was preceded by a strong hint, a weak hint or no hint at all. If they make a correct entry, then they probably know the requisite knowledge. This seems a bit unrealistic to us, especially when the last possible hint is so specific that it essentially tells the student what to enter (as often occurs in Andes, the Andersonian tutors, and many others). Perhaps the most elaborate and potentially accurate method of interpreting hints is used by the SMART ITS (Shute 1995), which uses a non-linear function derived from reports by experts, that boosts the level of mastery by different amounts depending on the specificity of the hint and the level of mastery before the hint was given.

Our approach attempts to be more principled by modeling a simple "theory" of hints directly in the Bayesian network. The theory is based on two assumptions:

- Hints from Andes' Procedural Helper are worded so as to remind the student of the requisite knowledge, rather than teach it. (Teaching missing pieces of knowledge is handled by the Conceptual Help system, which is not discussed here.) Thus, procedural hints do not directly cause students to master knowledge.
- A strong hint increases the chance that the student can guess the next action rather than derive it from her knowledge. Thus, a hint can cause an entry directly.

In other words, hints affect actions directly but not domain knowledge.

This mini-theory of hints is encoded in the Bayesian network as follows. Whenever a hint has been given for a node, a new node is attached to the network as its parent, representing the fact that a hint was given. The conditional probability table on the target node is modified so that the target node may be true if it was derived either via the application of a known rule, or via guessing based on the hint. Moreover, the higher the specificity level of the hint, the more likely that the target node is true (The specificy levels are the numbers that appear at the beginning of the hint templates shown earlier). In operation, this means that when the student makes the corresponding entry, the hint node "explains away" some of that evidence, so the probability of mastery of the requisite knowledge is not raised as much as it would be if the student made that entry without receiving a hint.

Evaluations of Andes

In the Fall semester of 1997, an earlier version of Andes was used in a formative evaluation by students in the introductory physics course at the US Naval Academy. About

160 students were asked to use Andes in their dorm rooms to do their physics homework for three weeks. Students were given a short pre- and post-test to assess the effect of using Andes on their understanding of physics concepts. Only 85 students ended up using the system enough to evaluate their test results. A multiple regression for these students, with post-test score as the dependent variable, shows a small but significant positive effect of the number of times a student asked for help ($p < .05$, $R^2 = .016$). The only other variable to have a significant effect on post-test score was the student's pre-test score.

The small size of the effect of asking for help, together with reports from students that the hints did not always seem relevant to what they were thinking about at the time they asked for help, led us to revise the plan recognition algorithm to its present form. In the version of Andes used in the first evaluation, the plan recognition strategy was simply to assume that the goal node with the highest probability in the entire network was the one the student was addressing. However, since there is no temporal information represented in the Bayesian network, this meant that the system was ignoring evidence about *when* actions had been done. The version of the procedural help system described in this paper addresses this problem by using the student's most recent action as the starting point in its search for the next hint target.

Preliminary results from 25 students who used the new version of Andes in the Spring semester show that the number of help requests per problem went up from 0.19 in the Fall to 0.52 in the Spring. Test results for these students are not yet available as of this writing.

As the project moves forward, we will continue to gather data from such formative evaluations. These evaluations are invaluable in both assessing the effectiveness of the system and suggesting new directions and improvements.

Future work and conclusions

There are several areas of future work planned for the Andes ITS. These include:

- Improved language generation: for instance, using discourse cues and more sophisticated surface generation to improve the coherence and grammaticality of the output.
- Tutorial planning: a new project (CIRCLE) is looking at the problem of deciding what kind of response to give to the student at any given time (e.g. a hint vs. an longer explanatory subdialog vs. no response).

In this paper we have presented a framework for generating responses to help requests that is particularly relevant to domains in which there is uncertainty about the user's mental state. We would argue that this uncertainty exists to some degree in most domains for which help systems are implemented. Our Procedural Help module performs three functions: it decides on the most effective topic for its help, it generates a hint about that topic taking into account the previous discourse context, and it updates its model of the user's mental state as a result of having received the hint. Furthermore, the integration of these abilities with a general knowledge assessment tool means that Andes can adapt its help as the student's level of knowledge changes over time.

Acknowledgements

The authors would like to thank Patricia Albacete, Zhendong Niu, Kay Schulze, Stephanie Siler, Bob Shelby, and Mary Wintersgill for their many ongoing contributions to this work. This research is supported by ARPA's Computer Aided Education and Training Initiative under grant N660001-95-C-8367, by ONR's Cognitive Science Division under grant N00014-96-1-0260, and by AFOSR's Artificial Intelligence Division under grant F49620-96-1-0180.

References

Anderson, J.; Corbett, A.; Koedinger, K.; and Pelletier, R. 1995. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences* 4(2):167–207.

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64:53–79.

Conati, C., and VanLehn, K. 1996. POLA: a student modeling framework for Probabilistic On-Line Assessment of problem solving performance. In *Proceedings of the 5th International Conference on User Modeling*.

Conati, C.; Gertner, A. S.; VanLehn, K.; and Druzdzel, M. J. 1997. On-line student modeling for coached problem solving using Bayesian networks. In *Proceedings of UM-97, Sixth International Conference on User Modeling*, 231–242. Sardinia, Italy: Springer.

Huber, M.; Durfee, E.; and Wellman, M. 1994. The automated mapping of plans for plan recognition. In *Proceedings of the 10th Conference on Uncertainty inArtificial Intelligence*, 344–351.

Jameson, A. 1995. Numerical uncertainty management in user and student modeling: an overview of systems and issues. *User Moldeing and User-Adapted Interaction* 3-4(5):193–251.

Martin, J., and VanLehn, K. 1995. Student assessment using Bayesian nets. *International Journal of Human-Computer Studies* 42:575–591.

Merril, D. C.; Reiser, B. J.; Ranney, M.; and Trafton, J. G. 1992. Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences* 3(2):277–305.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, Calif: Morgan Kaufman.

Pynadath, D. V., and Wellman, M. P. 1995. Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the 11h Conference on Uncertainty in Artificial Intelligence*, 472–481.

Shute, V. J. 1995. SMART: Student modeling approach for responsive tutoring. *User Modeling and User-Adapted Interation* 5:1–44.

VanLehn, K.; Jones, R. M.; and Chi, M. T. H. 1992. A model of the self-explanation effect. *The Journal of the Learning Sciences* 2(1):1–59.

VanLehn, K. 1996. Conceptual and meta learning during coached problem solving. In Frasson, C.; Gauthier, G.; and Lesgold, A., eds., *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems ITS '96*. Springer. 29–47.