International Journal of Artificial Intelligence in Education (2001), 12, 154-184

Bayesian student modeling, user interfaces and feedback: A sensitivity analysis

Kurt Vanlehn, *Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA, 15260; E-mail: vanlehn@cs.pitt.edu*

Zhendong Niu, *Carnegie-Mellon University, Pittsburgh, PA, 15213; E-mail: niu@cs.cmu.edu*

Abstract: The Andes physics tutoring system has a student modeler that uses Bayesian networks. Although the student modeler was evaluated once with positive results, in order to better understand it and student modeling in general, a sensitivity analysis was conducted. That is, we studied the effects on accuracy of varying both numerical parameters of the student modeler (e.g., the prior probabilities) and structural parameters (e.g., whether the tutor uses feedback; whether the tutor insists that students correct errors; whether missing entries are counted as errors). Many of the results were surprising. For instance: Leaving feedback on when testing students improved the assessor's accuracy; Long tests harmed accuracy in certain circumstances; CAI-style user interfaces often yielded higher accuracy than ITS-style user interfaces. Furthermore, we discovered that the most important problem confronted by the Andes student modeler was not the classic assignment of credit and blame problem, which is what Bayesian student modeling was designed to solve. Rather, it is that if students do not keep moving along a solution path, knowledge that they have mastered may not get a chance to apply, and thus the student modeler can not detect it. This factor had more impact on assessment accuracy than any other numerical or structural parameter. It is arguably a problem for all student modelers, and other assessment technology as well.

INTRODUCTION

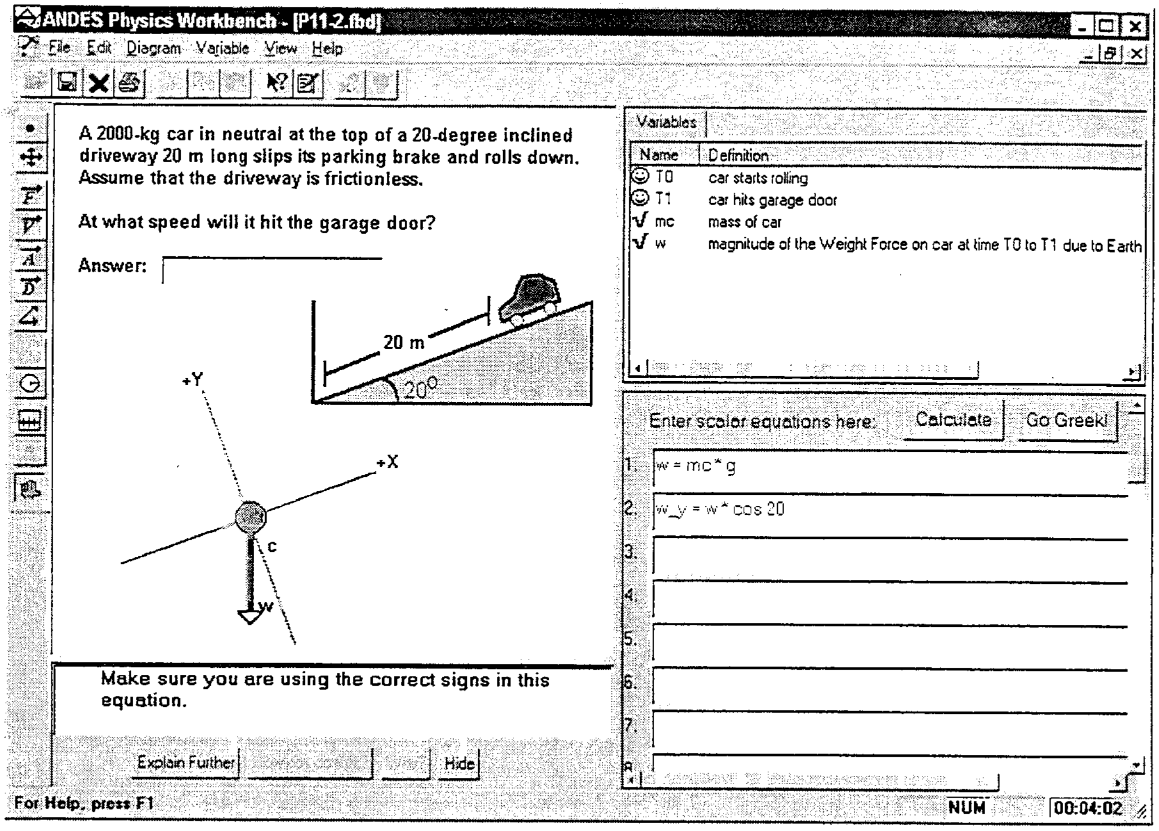
This paper reports on a sensitivity analysis of the student modeling module of the Andes intelligent tutoring system. First, Andes is described, then its student modeler is described, and finally the sensitivity analysis is described. Although the analysis was conducted with the student modeling module of a specific intelligent tutoring system (ITS), the conclusions turn out to apply to any assessment module. Moreover, they turn out to have implications for any performance assessment, where a performance assessment is a type of testing that includes data on the students' problem solving behavior as well as their answers (Linn, Baker, & Dunbar, 1991).

Andes: An intelligent tutoring system for physics

Andes (P.L. Albacete & K. VanLehn, 2000; Patricia L. Albacete & Kurt VanLehn, 2000; Conati, Gertner, VanLehn, & Druzdzel, 1997; Conati & VanLehn, 2000; Gertner, Conati, & VanLehn, 1998; Gertner, 1998; Gertner & VanLehn, 2000; Schulze, Shelby, Treacy, & Wintersgill, 2000; VanLehn, 1996; VanLehn et al., 2000) is a "homework helper" for students who are taking their first college course in physics. Andes is not intended to replace textbooks, lectures and the other course activities. It is intended only to help student's learn more from their problem solving.

Figure 1 shows Andes' user interface. The problem is displayed in the upper left window. Students can draw vectors and coordinate axes beneath it. In the lower right window, they can enter equations. In the upper right window, they can define variables. When Andes has something to say, it prints it in the lower left window.

As the students work, Andes gives them feedback and hints. In particular, every time the student makes an entry, such as a vector or an equation, Andes indicates whether it is correct or incorrect by changing its color. The default coloring is green if the entry is correct, and red if it is incorrect. Students are not required to correct their errors, although they usually do. This type of feedback is called *flag* feedback (Anderson, Corbett, Koedinger, & Pelletier, 1995), because the tutor only flags the error instead of insisting that the student correct it.


ANDES Physics Workbench - [P11-2.fbd]
File Edit Diagram Variable View Help
20 m
20°
+Y
+X
c
w
Variables
Enter scalar equations here:
Calculate
Go Greek!
Explain Further
Hide
For Help, press F1
NUM
00:04:02

A 2000-kg car in neutral at the top of a 20-degree inclined driveway 20 m long slips its parking brake and rolls down. Assume that the driveway is frictionless.

At what speed will it hit the garage door?

Answer:

Name	Definition
☺ T0	car starts rolling
☺ T1	car hits garage door
✓ mc	mass of car
✓ w	magnitude of the Weight Force on car at time T0 to T1 due to Earth

1. w = mc * g
2. w_y = w * cos 20

Make sure you are using the correct signs in this equation.

Figure 1. Andes' user computer interface

Students can ask for two kinds of hints. When an incorrect entry is selected, they can ask "what's wrong with that." They can also ask for a suggestion for what to do next. Andes, like most human tutors (Person, Kreuz, Zwaan, & Graesser, 1995), often answers obliquely because its primary goal is to get the students to learn physics, so helping them solve the problem efficiently is only a secondary goal. In fact, if Andes believes the request for a hint is due to a misunderstanding of a physics concept, it will give a minilesson on the concept instead of a hint. Minilessons consist of short hypertexts, often involving animations, that teach the concept both in its general form and in the current problem context(Albacete, 1999; P.L. Albacete & K. VanLehn, 2000; Patricia L. Albacete & Kurt VanLehn, 2000).

Andes has been used every semester since Fall 1997 in an introductory physics course at the US Naval Academy. These formative evaluations have been extremely important for developing the system and its surrounding pedagogy. A summative evaluation conducted in Fall of 1999 showed that Andes students learned significantly more than students who did the same homework problems on paper (Gertner & VanLehn, 2000). Data from a second summative evaluation conducted in Fall of 2000 are currently being analyzed (Shelby, Schulze, Treacy, Wintersgill, & Vanlehn, in prep.).

The Andes student modeler

When the student makes an entry, Andes has to determine whether the entry is correct so that it can give flag feedback. More importantly it must determine what line of reasoning the student used in producing it so it can give appropriate hints if asked. Such analyses of student entries are called *student modeling* in the intelligent tutoring system literature (the same process is called *user modeling* in the help systems literature).

Student modeling in Andes is done using a solution graph. Each problem has its own solution graph that records all possible correct entries and the lines of reasoning behind them. It is created by solving the problem with a rule-based problem solver and saving the logical dependencies as a directed acyclic graph (Russell & Norvig, 1995). The rules in the problem solver represent both general physics principles such as Newton's laws, and problem solving tactics such as which objects to apply the laws to. The rules are what we would like the students to learn.

In order to speed up student modeling during the tutoring session, the solution graphs are created in advance and stored in files. When a student makes an entry, Andes merely looks it up in the solution graph of the current problem. Because the solution graph contains all possible correct entries, the student's entry is flagged as correct if and only if it is found in the solution graph. Thus, Andes can give feedback nearly instantly.

If the student asks for a hint, Andes attempts to figure out what the student is trying to do so that it can construct a hint that is appropriate for the student's current goals. This process is called *plan recognition* or sometimes *model tracing*. It is a difficult AI problem because there can be multiple explanations for the student's recent behavior and not enough information to determine which one is most likely to be a correct account of the student's entries. This classic problem is called the *assignment of credit* problem (VanLehn, 1988).

Andes' solution to the assignment of credit problem is Bayesian. That is, it assumes that credit should be apportioned according to Bayes rule:

$$P(\text{Explanation} \mid \text{Evidence}) = \alpha\, P(\text{Evidence} \mid \text{Explanation})\, P(\text{Explanation})$$

where the constant α is adjusted so that certain probabilities sum to 1. The point here is that the amount of credit given to an explanation is proportional to how strongly it predicts the evidence and how likely it was before the evidence was observed. For instance, if X and Y are both equally strong predictors of the evidence, but X was more likely than Y before the entries were made, then X will get more credit for explaining the evidence than Y.

In principle, the Bayesian solution is the optimal way to assign credit (Pearl, 1988). In practice, it requires parameter values such as P(Explanation | Evidence) and P(Explanation) that are difficult to obtain empirically, so they are often estimated ,which makes them suspect. Nonetheless, the Bayesian solution to the assignment of credit problem at least confines the area of suspicion to the numerical parameters. With a heuristic solution to the assignment of credit problem, both the algorithm and the parameters are suspect.

The structure of Andes' Bayesian network

Andes implements a Bayesian solution to the assignment of credit problem by converting the solution graph to a Bayesian network. Bayesian networks are described in most recent AI textbooks (e.g., Russell & Norvig, 1995), so we will not explain them here. However, we will explain the particular Bayesian networks that Andes uses.

An Andes Bayesian network has three kinds of true/false nodes:

- A *rule* node is true if the student has mastered the rule; that is, the student always applies the rule whenever it is appropriate to do so.
- A *proposition* node is true if the proposition is in the student working memory. Propositions are either problem-specific facts (e.g., "there is a tension force on block2 exerted by string1") and goals (e.g., "to draw forces on block2").
- A *rule application* node is true if the rule application has occurred. For instance, suppose the rule "If ?string is tied to ?body, then there is a tension force on ?body due to ?string." can be applied to the fact "string1 is tied to block2." This rule application

would produce the fact "there is a tension force on block2 due to string1." A node represents the rule application. It the application has occurred, then the node is true. If it has not occurred, then the node is false.

The nodes are linked together to record the reasoning that is involved in solving the problem. Figure 2 shows a fragment of an Andes Bayesian network. Proposition 1 is given in the problem statement. Rule 1 matches it and produces proposition 2, so rule application 1 has both proposition 1 and rule 1 as parents, and has proposition 2 as a child. Rule 2 matches proposition 2 and proposition 3. Applying it produces proposition 4. Rule application 2 records this. However, proposition 4 has a second derivation. It can also be derived by applying rule 3 to proposition 3 and 5. In short, this network represents the firing of 3 rules. It usually takes around a hundred rule applications to solve a simple physics problem.

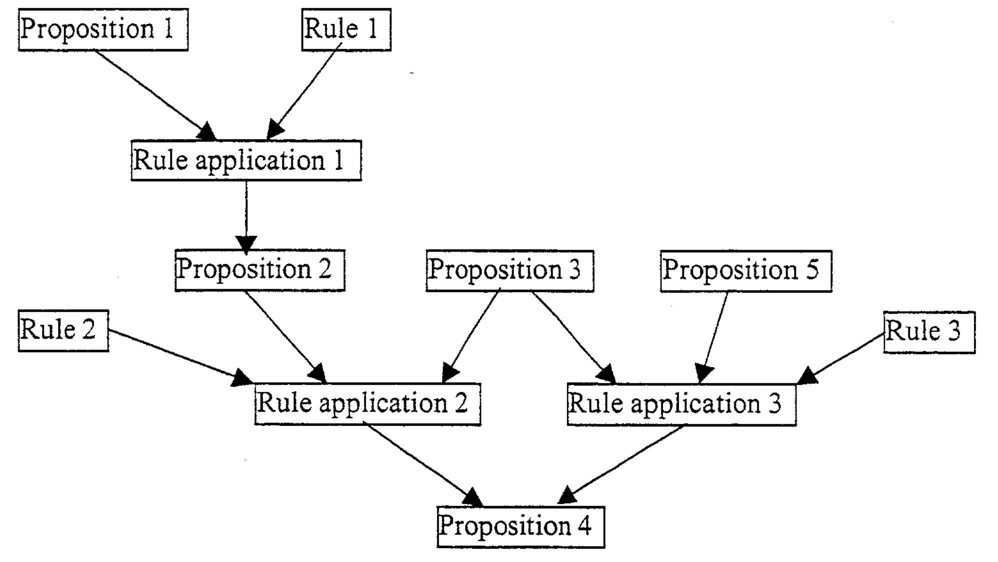


Figure 2. An Andes Bayesian network

In addition to rule, proposition and rule application nodes, Andes has *strategy* nodes, which are used when there are two mutually exclusive approaches to solving a problem. For instance, it usually does not matter how the coordinate axes are oriented. A standard orientation is to have the x-axis be horizontal. However, in some problems, one may wish to make it parallel to a non-horizontal vector. In such problems, a strategy node would be used to record the fact that if the solver uses one orientation, then the other orientation cannot also be used. They are mutually exclusive. Strategy nodes are relatively uncommon in Andes networks, so the details will not be described here (see Conati et al., 1997).

Every node has a conditional probability table associated with it. For nodes without parents, the table represents the prior probability of the node. For instance, rule nodes do not have parents, so their conditional probability tables consist of a single probability, which indicates the probability that the student has mastered that rule before beginning to solve this problem. If a proposition has no parents, then it is a fact or goal that is mentioned in the problem statement. We give it a high prior probability (.95 for fact; .98 for goals) to indicate that the student probably put that proposition in working memory. The other proposition nodes and the rule application nodes have parents, so their conditional probability tables are more complex.

Table 1. Conditional probability table of a rule application node

Rule	Prop 1	Prop 2	P(occur)	P(not occur)
T	T	T	0.999	0.001
T	T	F	0	1.0
T	F	T	0	1.0
T	F		0	1.0
F	T	T	0	1.0
F	T	F	0	1.0
F	F	T	0	1.0
F	F	F	0	1.0

All rule application nodes have the same kind of conditional probability tables. They represent that a rule application occurs only if all the propositional parents are in working memory and the rule is mastered. Table 1 shows one for a rule that has two propositional parents, such as rule application 2 in Figure 2. If the rule is mastered (T in the first column) and both propositions are present in working memory (T in the second and third column), then the rule application occurs with probability 0.999. If the rule is unmastered (F in the first column) or either proposition is absent, then there is zero probability that the rule application will occur. Thus, the conditional probability table for a rule application represents a logical AND, except that there is a small probability (0.001) that even if all the parents are true, the rule application will still not occur. This probability (0.001) is called the *slip* parameter, because it represents the chance that a student will make an unintentional error of omission. Currently, all Andes rule application nodes have a slip parameter value of 0.001.

Table 2. Conditional probability table of a proposition node

RA1	RA2	P(in WM)	P(not in WM)
T	T	1.0	0.0
T	F	1.0	0.0
F	T	1.0	0.0
F	F	0.2	0.8

The conditional probability table associated with a proposition reflects the assumption that a proposition is put in working memory either if it is derived by a rule application or the student makes a lucky guess. Table 2 shows the table for a proposition that has two derivations, and thus two rule applications as parents (i.e., like Proposition 4 of Figure 2). If either or both of the rule applications have occurred, the proposition is definitely (P=1.0) in working memory. If neither occurred, there is a 0.2 chance of a lucky guess. This probability (0.2) is called the *guess* parameter. Different propositions have different values for their guess parameters, because some propositions are easier to guess than others are. For instance, guessing whether acceleration is up or down is easier than guessing the algebraic formula for its magnitude. Currently, about 74% of the Andes nodes have a guess parameter value of 0.2, 25% have a value of 0.1 and 1% have a value of 0.01.

Most propositions have only one derivation. In this case, when Andes builds a Bayesian network from the solution graph, it combines the node for the proposition and the node for the rule application that produces it. For instance, in Figure 2 it would combine Rule application 1 with Proposition 2. The conditional probability table of such a combined node looks just like that of Table 1 except that the zero probabilities are replaced with the proposition's values for the guess parameter.

How the Bayesian network is used

The network's main use is determining which rules the student has probably mastered. However, those probability evolve over time, so Andes uses a dynamic belief network (Russell & Norvig, 1995, section 17.5). Essentially, Andes processes each problem separately while keeping an evolving record of the rule nodes' probability. More specifically, Andes maintains a file on each student that contains, for each rule, the probability that the student has mastered that rule. When a problem is opened and its Bayesian network is created, Andes initializes the prior probabilities of the rule nodes to the contents of this file. As the student solves the problem, the network is updated and the posterior probabilities of the rules change. When the problem is closed, the posterior probabilities are stored in the student's file. These new values represent the probability of mastery given all the evidence that has been gathered up to and including the problem that was just closed.

Actually, the probabilities in the file represent only the marginal probabilities of the rules and not the conditional dependencies among them. This may create some inaccuracy in the assessment. Techniques have been developed to reduce this potential inaccuracy (Martin & VanLehn, 1994), but they can be computationally expensive. They are not used in Andes, and they are not commonly used in other dynamic belief network applications.

When Andes has initialized the prior probabilities of the rule nodes, and the student has solved the problem thus producing a set of entries, the Bayesian network is updated in the following way. First, each student entry is matched to propositions in the Andes network. A correct entry always matches at least one proposition and often matches multiple propositions. For instance, a tension force vector drawn by the student will match propositions stating that the tension force exists, that it acts on a certain body and that it has a certain direction. Incorrect entries usually do not match any propositions.

The propositions that are matched are clamped True. This evidence is propagated through the network by an update algorithm. Andes prefers to use an exact algorithm (the Lauritzen-Speigelhalter algorithm: Pearl, 1988), but uses an approximate algorithm (the likelihood sampling algorithm: Fung & Chang, 1989; Shachter & Peot, 1989) if the network is too big for its preferred algorithm. When the update is done, all the nodes have posterior probabilities reflecting the entries made by the student so far.

Andes uses the posterior probabilities as it constructs responses to the student's requests for help (Gertner et al., 1998). For instance;

- it uses the posterior probabilities of the goal proposition to determine what the student is probably intending to do—an essential first step in constructing a useful hint.
- If the student has made an error, Andes answers a "what's wrong with that" request by first determining the most likely explanation for the error.
- If the probability of mastery on an important rule is particularly low and the rule seems to be causing the error, then Andes will give a minilesson on the rule instead of a hint.

In short, the student modeling system in Andes consists of two major modules. A rule-based problem solver solves physics problems in all possible ways and creates a solution graph for each. A second module converts the solution graph to a Bayesian network, clamps nodes in the network that match student entries, and determines the posterior probabilities of the nodes in the network. The latter module is called the *assessor.*

Evaluation of the assessor

A simple method for evaluating a student modeling system is to develop a model of a student, use it to predict an exam score, then see if the prediction is accurate. Anderson et al. (1995) used the student modeling component of their Lisp tutoring system to predict midterm and final exam scores from courses taught by the tutor. Shute (1995) has used the same method for evaluating her SMART student modeling system. Both investigations found strong correlations between predicted and actual exam scores.

Although simple, this method is incomplete in several ways. First, the student modeler assesses the student's competence on many small pieces of knowledge, yet all these predictions are reduced to a single number, the exam score. Such a crude evaluation may allow defects in the student modeler to go undetected. Second, this method confounds an evaluation of the assessor's algorithm with an evaluation of the domain knowledge used by the problem solver which may or may not be an accurate analysis of the task domain. Lastly, the traditional method yields just a simple correlation. If the correlation is low, it does not say *why* the student modeler performed poorly or how to improve it. For these reasons, we sought a different evaluation than the standard one.

We used several novel methods to evaluate Olae (Martin & VanLehn, 1993; Martin & VanLehn, 1995a, 1995b; VanLehn & Martin, 1998), a predecessor of Andes that used the same student modeling technique. For instance, one evaluation was to have a human assessor analyze verbal protocols from several physics students who were solving a large number of problems. Olae also analyzed the protocols, but had access to only the student's written actions and not their verbal utterances. Thus, the human assessor's judgment of which rules were mastered could be considered a "gold standard" against which Olae's student model could be compared. The bottom line was that Olae fared quite well on all the evaluations we conducted (VanLehn & Martin, 1998).

However, we still felt that we did not thoroughly understand why our technique was doing well, nor what could be done to make it better. Moreover, we weren't sure whether its performance was due to its Bayesian modeling technique or something else. We were also worried about its sensitivity to its parameters: the guess parameter, the slip parameter and the prior probabilities on the rules. Lastly, Olae used a small physics knowledge base and an awkward user interface. We needed to evaluate our techniques in the scaled-up environment of Andes. These concerns led to designing a new evaluation.

DESIGN OF THE SENSITIVITY ANALYSIS

In order to more deeply understand the properties of our technique, we conducted a sensitivity analysis. That is, we varied parameter values and structural features of the assessor in order to determine which ones made the most difference in the accuracies of its assessments.

Because a sensitivity analysis requires repeated analyses under many different conditions, the only practical way to do it is with simulated students. A simulated student is a computer model of a human student (e.g., VanLehn, Ohlsson, & Nason, 1994). Although it is only an approximation of human behavior, we know exactly what its competence is. This makes it easy to determine if the assessor's predictions of the simulated student's competence are accurate. Moreover, one can easily generate simulated students with known characteristics, such as a set of students with low knowledge or a tendency to guess frequently. This feature is nearly essential when conducting a sensitivity analysis, because one wants to understand how the assessor will behave under different conditions. Lastly, the simulated student and the assessor use exactly the same representation of domain knowledge, so unlike evaluations with human students, evaluations with simulated students do not confound accuracy in representing the domain knowledge with accuracy of the assessor's algorithms.

One way to implement a simulated student is to use the Andes problem solver, after modifying it slightly to model slips and guesses. A simulated student is generated by randomly deleting rules, thus modeling the fact that different students have different knowledge. The reduced knowledge base is used to solve problems. Recordings of the solver's actions while solving the problems are given to the assessor. The assessor should assign the deleted rules a low posterior probability of mastery, and the other rules a high probability.

In order to simplify the analysis, we concentrated only on the assessors' predictions of rule mastery. The assessor also predicts whether a simulated student will have goals and other propositions in working memory, but we did not evaluate those predictions. Thus, we concentrated only on its assessment of long term knowledge (knowledge tracing) and not on its ability to do plan recognition (model tracing).

The first section describes the overall format of the evaluation. The second section describes exactly how simulated students are generated and used to solve problems. The third section describes how the assessor generates predictions and how the accuracy of those predictions is measured.

Overall format of the evaluation

The sensitivity analysis consisted of large number of "runs." Each run evaluated the assessor with a different setting of the simulated student parameters and the assessor parameters.

Each run began by generating 30 simulated students. Each student had a different randomly generated set of mastered rules. The rules in the knowledge base that are not marked as mastered were called unmastered. Each simulated student solved 23 physics problems, and its work was saved in a log file. The log file was given to the assessor, which determined a posterior probability for every rule. These were used to determine the accuracy of the assessor.

The 23 problems used in the evaluation sampled a range of topics and difficulties. Some were simple one-dimensional kinematics problems, such as "A modern supertanker can chug along at 16 knots (8.33 m/s). It takes 1200 seconds to bring it to a full stop. Calculate the corresponding deceleration in m/s^2." Some of the most complex problems required applying both Newton's law and kinematics in two dimensions, as in the problem of Figure 1. The problems were taken from 3 chapters in a physics textbook. We used only problems that could be analyzed with the exact Bayesian network update algorithm, because we did not want the random variations introduced by the approximate algorithm to obfuscate the results.

The Andes assessor is incomplete in at least one respect: It assumes the student's knowledge does not change during the assessment period. That is, it assumes that learning and forgetting do not occur. To rectify this oversimplification requires giving recent evidence more weight than older evidence. One technique is to use a moving window (e.g., Gitomer, Steinberg, & Mislevy, 1995) that counts only the most recent N pieces of evidence. Another is to use the two-state Markov model of Corbett and Anderson (1995). Before we tackle the issue of developing an assessment of changing knowledge, we first want to understand how well our assessment of constant knowledge works. Thus, we designed the simulated students so that they did not learn or forget during the course of solving the 23 problems. That is, a simulated student uses the same set of mastered rules on every problem.

Generating the simulated students and the log files

Having presented the overall format of a "run," it is time to get into some of the details. First, the method of generating simulated students and log files is described.

Although it is straightforward to generate simulated students by deleting rules, one must run the solver once for each student and each problem, so we used a more efficient but logically equivalent implementation. We solved each problem once and generated a Bayesian network for it. To generate an individual simulated student, we randomly marked some rules as mastered. This was controlled by a competence parameter associated with each rule. For instance, a value of 0.75 for a rule meant that on average, 75% of the simulated students would have that rule marked as mastered.

The next step was to generate a log file representing the student's work on the 23 physics problems. Given the network for a particular problem, we clamped the mastered rules True and the unmastered rules False. The propositions given in the problem statement were clamped True. After the network was updated, the posteriors on the propositions indicated which ones would probably be in working memory. For instance, if a proposition had only a single derivation via a rule application whose rule was unmastered, then the proposition typically had a low posterior probability (in the vicinity of 0.2) indicating that it would only be true if the student made a lucky guess.

Log files were generated so that the chance of a proposition appearing in the file was the proposition's posterior probability. Thus, propositions with the low posteriors typical of guessing got into the log file only rarely, whereas propositions with the high posteriors typical of derivation via mastered rules almost always got into the log file.

Running the assessor

To process a log file, the assessor began by analyzing the first problem. It assigned prior probabilities to the rules and clamped the nodes identified in the log file to True, then ran the update algorithm. This gave a posterior probability for each of the rules. These were then "rolled up" by transferring the rule's posteriors from the first problem to the rule's priors for the second problem. The assessor then clamped log file nodes for the second problem, updated the network, and rolled up the probabilities again. This cycle continued until all 23 problems had been processed.

The results were that every rule had a probability of mastery reflecting how often it appeared to have been used during the problem solving. If the assessor was accurate, then the rules that were mastered in the simulated student (henceforth called "mastered rules") should have high posteriors, while the unmastered rules should have low posteriors.

One way to check this is to make histograms of the posteriors, such as the ones shown in Figure 3.

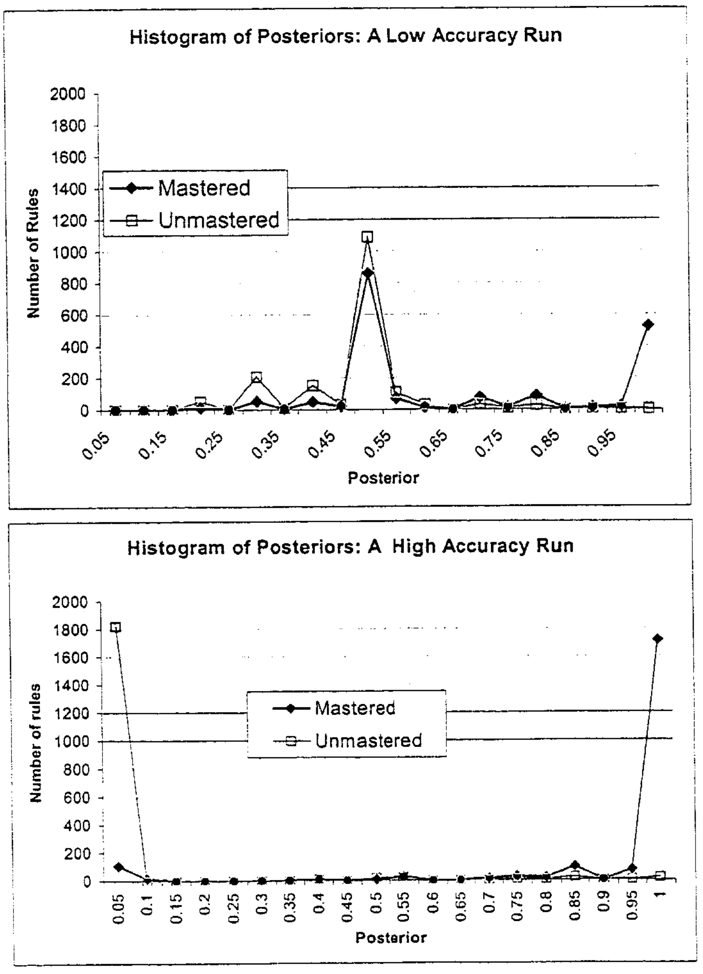


Figure 3. Histograms of posterior probabilities of rules

Although each of the 30 simulated student has the same rules (approximately 140), the same rule in different simulated students will usually have different posteriors at the end of the run. Thus, there are around 30*140=4200 rules whose mastery has been assessed. The histograms of Figure 3 show how many of the 4200 rules (y-axis) had each posterior (x-axis).

The top graph shows a histogram for a run whose accuracy turned out to be quite high. Notice that the histogram for the mastered rules has a single peak near 1.0. That is, most of the mastered rules had high posteriors. The unmastered rule's histogram peaks near 0, indicating that most of the unmastered rules had low posteriors. The bottom graph of Figure 3 shows a low accuracy run. The histogram of the mastered rules has two peaks. One is near 1.0 and the other is near 0.5. The unmastered rules have just one peak near 0.5. However, what is remarkable is that the shapes of the two histograms are nearly identical. Clearly, the assessor is not doing a good job of differentiating mastered from unmastered rules.

Although the histograms are revealing, they do not make it easy to determine how much better one run is than another. It is much more convenient to have a numerical measure of accuracy. A simple one is just to count how many times Andes makes a correct prediction. However, Andes does not produce a definite mastered/unmastered prediction that could be easily compared to the true mastered/unmastered state of the rule. Instead, it produces only a posterior probability. Thus, we use a threshold on the posterior probability to convert it to a binary value. That is, given a particular threshold T, if a rule's posterior is above T, then it is predicted to be mastered. If the posterior is less than or equal to T, then it is predicted to be unmastered. Although there are other methods for comparing a binary value to a continuous one, using thresholds corresponds to how the posteriors are typically used in tutoring systems: When all of the posteriors are above a certain threshold, the student is allowed to go on to the next unit in the curriculum. For instance, the PACT tutors all use a threshold of 0.95 (Albert Corbett, personal communication, 1998).

The threshold-based comparison allows us to use tables such as Table 3. In this illustrative table, the 30 simulated students had a 3735 mastered rules and 375 unmastered rules (i.e., each student had about 140 rules, most of which were mastered). This table uses a threshold of 0.95. Of the mastered rules, Andes assigned probabilities that were above threshold to 3329 rules. Of the unmastered rules, it assigned probabilities that were below threshold to 61 rules.

Table 3. Number of rules in a set of simulated students

	Above threshold	Below threshold	Total
Mastered	3329	406	3735
Unmastered	314	61	375

We used two measures for most of our analyses:

- *Accuracy on mastered rules.* This measures how well the assessor does at identifying the simulated student's mastered rules. It equals the number of mastered rules that are above threshold divided by the number of mastered rules. For this illustration, the mastered accuracy is 3329/3735 = .89.

- *Accuracy on unmastered rules.* This measures how well the assessor does at identifying the simulated student's unmastered rules. It equals the number of unmastered rules that are below threshold divided by the number of unmastered rules. For this illustration, the unmastered accuracy is 61/375=.16.

The advantage of using these two numbers is that many of the parameter variations affected only one of them. If we were to use a composite of the two kinds of mastery, then it would be harder to interpret the results.

Nonetheless, after the results have been analyzed with respect to these two types of accuracy, a single measure of accuracy is needed in order to determine which versions of the assessor are the best overall. A later section discusses traditional measures and selects the most appropriate one.

Parameters varied

Many variations of the assessment could be studied. After an initial exploration, we settled on studying all possible combinations of 2 numerical parameters of the simulated students, 2 numerical parameters of the assessor and 4 structural parameters that affect both the simulated students and the assessor. Subsequent paragraphs describe each parameter and how it was varied (see Table 5 for a summary).

Simulated student competence. As described earlier, each rule in a simulated student has a parameter that determines how likely that rule is to be marked mastered. We used two different values for these competence parameters. "*High*" competence students had a value of 0.9 for every rule's competence parameter, which meant that on average, that rule would be mastered in 90% of the simulated students. "*Low*" competence students had competence values drawn from a study of human students (VanLehn, Niu, Siler, & Gertner, 1998). In that study, students who were about to use Andes were given a physics test designed to measure mastery of individual levels. This allowed estimation, for each rule, of how many students had mastered it and how many had not mastered it. The ratio was used as the competence value for that rule when generating a low competence student. For instance, if 75% of the students appeared to have mastered a certain rule on the test, then the rule was assigned a competence value of 0.75. Unfortunately, the test could not assess advanced physics knowledge because the students were just starting their study of physics. Hence, 60% of Andes' rules did not appear on the test. These were assigned a competence value of 0.5.

Simulated student guessing. The log files were generated with a Bayesian network that had the same format as the one used in the assessor. Thus, they had guess parameters for each rule application. We used two values for these parameters. *No-guess* simulated students never guessed. That is, in the Bayesian networks used to represent the simulated students, the guess parameter of every rule was set to zero. The *guessing* students had their guess parameters set as they are usually set in Andes.

Assessor guessing. The guess parameters in the assessor's networks were either given their normal Andes value or 0.00000001 (they could not be set to zero for technical reasons).

Threshold. A threshold is used to determine whether the posterior probability of a rule was high enough to consider it mastered. We used either 0.95 or 0.80.

Observability. The Andes user interface allows only about 80% of the propositions to be directly observed. When running a simulated student, only observable propositions are entered in the log file. For instance, the Andes user interface provides no way for students to enter their goals, so no goal nodes are placed in the log files. However, we were interested in what would happen if the user interface were enhanced so that the students could tell Andes about the goals they were working on and other propositions that are not normally observable. Thus, we varied whether only *some* propositions were marked observable or *all* propositions were marked observable.

Negative evidence. When working on paper, students are usually not required to enter any particular notations except the answer. Which vectors they draw and which equations they write are up to them. If they can do the whole problem in their head, then entering only the answer is fine. Because the Andes interface is intended to be as much like paper as possible, it also does not force students to enter anything but the answer. Consequently, when a relevant entry is not made, we cannot assume that the student is unable to make it; the student may simply have chosen not to write it down. This creates an asymmetry. When there is a entry corresponding to a proposition, the proposition is clamped True in the assessor, thus representing that it is in the student's working memory. When there is no entry corresponding to a proposition, it is not clamped False as one might expect, but is instead left unclamped. Although the entry is missing from the student's work, the proposition may or may not be in the student's working memory. There are only a few exceptions to this. For instance, if a student enters a vector that points up when its correct direction is down, then because the student undoubtedly believes that vectors only have one direction, the student must believe that the vector does not point down. Thus, Andes clamps False the correct proposition, which is that the vector points down. In Bayesian parlance, clamping nodes False is called negative evidence. So Andes rarely uses negative evidence. Most of the evidence is positive (nodes clamped True).

This is not the only way to design a user interface. Many CAI systems require students to make certain entries. For instance, they might explicitly ask the student to draw, say, the gravitational force vector of an object. If the student does not make that entry, then the system knows that the student cannot make it. If the system used Bayesian networks, it could clamp the corresponding correct propositions False.

Thus, we studied two conventions. The *positive-evidence only* convention is the one used by Andes. If an observable propositional node occurs in the log file, it is clamped True; otherwise it is left unclamped. The *negative-and-positive evidence* convention is an idealization of the CAI-style interface: If an observable proposition is in the log file, it is clamped True, and otherwise it is clamped False.

Feedback. When using a tutoring system to assess performance rather than to teach, one's first inclination is to turn off the feedback and hints. For instance, Olae used no feedback and hints. However, we were interested in what would happen if feedback and hints were allowed. In fact, we were interested in evaluating the effects not only of Andes' flag feedback, but also of the PACT tutors' most common form of feedback, wherein students must correct an error before they are allowed to continue. With both flag feedback and mandatory correction feedback, students can ask for as much help as they want, and typically the last help message tells them exactly what to enter. The philosophy here is that if the students really can't make the entry despite all the scaffolding provided by the hints, then what they really need is a dose of modeling: they need to see what the right entry is. Instead of thinking of this convention as "mandatory correction feedback," one could think of it as "just in time examples" (Koedinger, personal communication, 1999).

At any rate, our simulated students clearly could not understand the natural language hints that Andes gives (or its minilessons!), so we had to simulate the effects of the two kinds of feedback. We modeled them by changing the conditional probability tables of the rule application nodes in the simulated students.

Table 4. The conditional probability tables of rule application nodes showing P(occur) for each type of feedback

Rule	Observable proposition	Unobservable proposition	No feedback	Mandatory correction	Flag feedback
T	T	T	0.999	0.999	0.999
T	T	F	0	0	0
T	F	T	0	0.999	0.7992
T	F	F	0	0	0
F	T	T	0	0	0
F	T	F	0	0	0
F	F	T	0	0	0
F	F	F	0	0	0

If mandatory correction feedback is used, then even though a student might make a mistake the first time they try to produce an entry, the entry will eventually be corrected. Thus, subsequent rule applications that input that entry will get a correct value instead of an incorrect one. To model this, we continue to let the truth value of a proposition represent the value that the student would originally enter before it is corrected by feedback. We do not explicitly represent the value of propositions after they are corrected because we know that it is always correct (i.e., True). Instead, we modify the conditional probability tables of all rule applications that take the proposition as input so that they assume that the proposition's value is correct. Table 4 shows a conditional probability table for a rule application that has two propositions as inputs, where one is observable and the other is not. The "no feedback" column shows the standard values for the probability of the rule application being True; they are the same as those in Table 1. The "mandatory correction" column shows the new values. When the rule is unmastered or the non-observable proposition is false, the rule application does not occur. Otherwise, it does occur, unless there is a slip (the slip parameter's value is 0.001).

In order to model the effects of flag feedback, we took advantage of an empirical finding. Both in Anderson et al.'s (1995) study of flag feedback and in our analyses of Andes log files from human students, it was found that students correct about 80% of the errors that the tutor flags. Thus, we assumed that when an observable proposition is false, there is an 0.8 chance that it will be corrected before it is used in a subsequent rule application. Thus, the conditional probabilities in Table 4 include 0.999*0.8=0.7992 in the "flag feedback" column.

Multiple derivations. All the structural variations mentioned so far—observability, negative evidence and feedback—correspond to changes in the tutor. We also wondered what would happen if the task domain were slightly different. In particular, a notorious problem in student modeling is the assignment of credit problem: when a true proposition has multiple derivations, which derivation should be given credit for its truth? To see what would happen if there were no assignment of credit problem, we edited the solution graphs to remove multiple derivations. In particular, whenever a proposition had multiple parents, we randomly removed all but one.

In summary, we explored the sensitivity of the assessor to the variations shown in Table 5. There are several parameters that are conspicuous by their absence. Our reasons for not including them require explanation.

Table 5. Parameters varied during the sensitivity analysis

Parameter	Values
Simulated students guessing	Andes' values; 0
Simulated students competence	Frequencies of mastery among human students; 0.9
Assessor guessing	Andes' values; 0.00000001
Threshold	0.95; 0.80
Observability	Andes' values; All propositions observable
Negative evidence	No (Andes value); Yes
Feedback	None; Flag (Andes' value); Mandatory correction
Multiple derivations	Yes (Andes' value); None

Prior probabilities. The prior probabilities of the rules in the assessor seem to be particularly important parameters. If two rules are competing to explain a true proposition, then the one with the higher prior probability gets more credit. In our preliminary studies, we were surprised to find that varying the priors actually made almost no difference in accuracy, except in one respect: The ratio between the prior probabilities and the guess parameters made a difference in the assessor's accuracy. If the priors were much higher than the guess parameter, then the rules got most of the credit and their posteriors were high. If the priors were low with respect to the guess parameter, then the rules got little credit and their posteriors were low. This makes sense, because the rules and the guesses compete to explain true propositions. Thus, we varied only the guess parameter's value, and not the priors. The priors were given the values obtained from our study of human subjects (VanLehn et al., 1998).

The slip parameter. In the assessor, there are only 3 kinds of numerical parameters: the priors, the guess parameters and the slip parameters. We've discussed the first two. Why was the slip parameter not included in the sensitivity analysis? The slip parameter represents the probability that a rule application will not occur even though all its parents are true. If a proposition is false, it could either be because none of its rule applications could occur due to false parents, or all of its rule applications had a slip. That is, slips and rules compete to explain false propositions. However, the assessor normally is not told when propositions are false. Propositions are clamped False only when negative evidence is being used. Thus, the slip parameter has a significant effect on the assessor's predictions only when negative evidence is being used. Thus, we did not think it worth the computational effort necessary to explore its affects on the assessment.

Although we planned to do a run of the assessor for each of the 384 combinations of the parameter values shown in Table 5, some of the parameters turned out to have such small effects that we did not carry them all the way through. Those parameters are discussed in the next section. The subsequent section presents the impact of the remaining parameters on the assessor's accuracy on *mastered* rules, and the third section presents their impact on the accessor's accuracy on *unmastered* rules. A final section discusses how to combine the two measures of accuracy, then presents the impact of the parameters on that measure of overall accuracy.

PARAMETERS WITH SMALL EFFECTS

Removing multiple derivations should have caused accuracy to go up, because it removes the assignment of credit problem. However, in all runs where multiple derivations were removed, there was a negligible change in accuracy. We made 28 runs with multiple derivations removed and various combinations of the other parameters. In all cases, the change in accuracy was ±2% or less. At first, this result amazed us. The problem of apportioning credit and blame has been central to the student modeling field. However, we then analyzed the networks and found that less than 3% of the propositions have multiple derivations. That is, in this part of physics, there is usually only one way to derive a proposition. When we discovered this, we did not bother to complete the remaining 36 runs that would be required to test the multiple derivations with the remaining combinations of parameters.

Both the simulated students and the assessor have guess parameters and we varied both independently. However, we found that the guess parameter in the assessor had little effect. In general, accuracy was slightly higher when the value of the guess parameter matched the value used in the simulated students. Thus, in all the analyses reported later we report values only from runs where the values of the guess parameters matched. In practice, this means that the guess parameter of the assessor would have to be adjusted empirically to approximate the guessing that occurred in the target population.

ACCURACY ON MASTERED RULES

Figure 4 graphs the accuracy on mastered rules for all of the runs of the assessors. That is, the y-axes show the proportion of the mastered rules that the assessor decided were mastered because their posterior probability was above the threshold T. The top pane of Figure 4 shows the result of assessing simulated students who did not guess; the middle pane shows the result of assessing students who did guess. Both low competence and high competence students are shown, each with two different thresholds for mastery. The bottom pane shows the average over all 8 data points from the first two panes. The x-axis of the graphs in Figure 4 are ordered so that the overall accuracy (lowest pane) is monotonically increasing.

Table 6: Abbreviations for assessor types

First letter: Feedback	Second letter: Observability	Third letter: Evidence
N = no feedback	A = all nodes observable	P = positive evidence only
F = flag feedback	O = only some observable	N = negative and positive
C = mandatory correction		

The abbreviations along the x-axes are explained in Table 6. They are used in many subsequent figures and discussions. For example, CON means that the assessor used mandatory correction, only some of the propositions were observable and observable propositions that were false (or true) in the simulated student were clamped False (or True) in the assessor.

Although the graphs shown in Figure 4 seems complex, there are really just three main trends that determine their shape:

1. There is a jump in the lowest graph that indicates that the first 8 assessors (NON through NAP) have considerably less accuracy than the last 4 assessors (FAN through CAP). That is, assessors with some kind of feedback (F prefix for flag feedback or C prefix for mandatory correction feedback) and all nodes observable (A in the middle) have significantly higher accuracy than any of the other assessors.

2. Within the group of 8 assessors (NON through NAP in the lowest graph), the ones with an N suffix are always less accurate than those with a P suffix. Within the group of 4 assessors (FAN through CAP), those with N suffixes are always less accurate than those with a P suffix. Thus, assessors that get negative as well as positive evidence (N suffix) have less accuracy than assessors that get only positive evidence (P suffix).

3. The assessors with low (0.8) thresholds have higher accuracy than the assessors with high (0.95) thresholds.

The subsequent sections discuss the reasons behind each of the findings. The point here is not just to understand which parameters cause variations in accuracy, but to understand *why.*

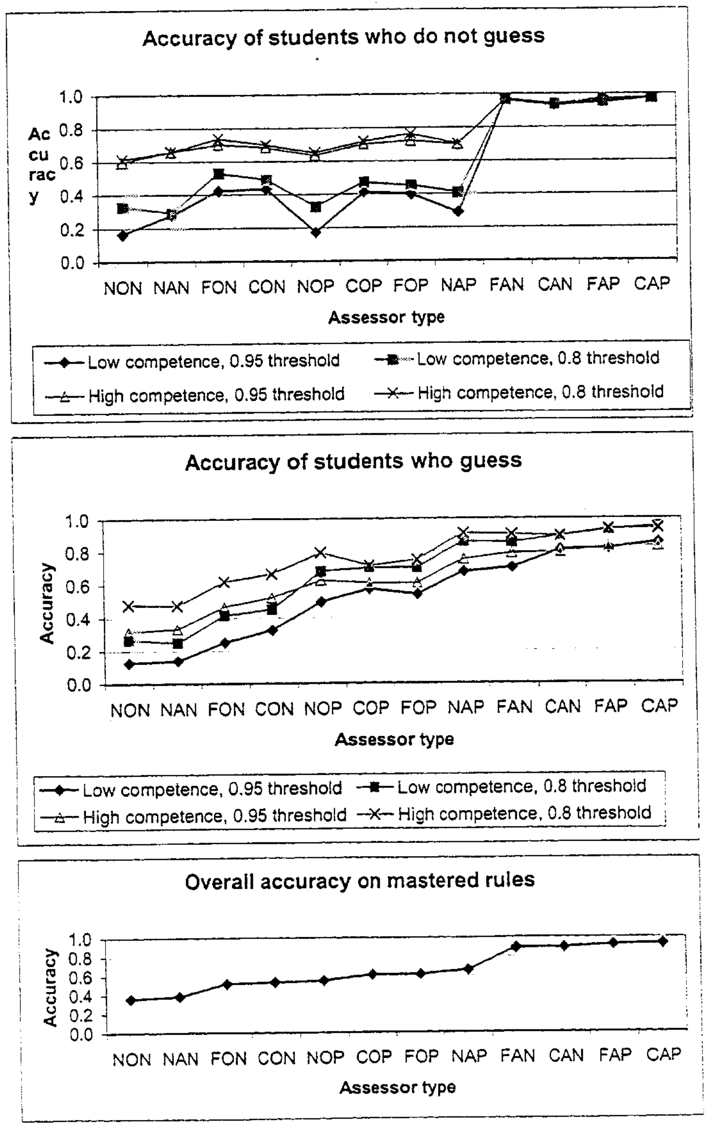


Figure 4. Accuracy of the assessment of mastered rules

Lowering the threshold increases accuracy on mastered rules

The assessors with 0.80 threshold generally had higher accuracy than those with on 0.95 threshold. This makes sense. Because accuracy on mastered rules is the percentage of mastered rules whose posteriors are *above* the threshold, lowering the threshold increases the number of rules above threshold, thus increasing accuracy.

Unfortunately, lowering the threshold is not a panacea, because doing so hurts accuracy on the unmastered rules. Because accuracy on the unmastered rules is the percentage of unmastered rules whose posteriors are *under* threshold, lowering the threshold reduces unmastered accuracy. Thus, one must set a threshold that is a compromise between the two types of accuracy.

Keeping students moving along a solution path improves accuracy at detecting mastered rules

Figure 4 indicates that accuracy in assessing mastered rules is best when there is some kind of feedback (F for flag feedback, or C for mandatory correction feedback) and all propositions are observable (A in the middle). The four assessors with these properties (FAN, CAN, FAP and CAP) produced nearly perfect accuracy when there was no guessing, and high accuracy when there was guessing. Moreover, the effect was quite large, indicating that this is an important finding to understand.

If we think about this in terms of real students instead of simulated ones, the finding makes perfect sense. The improvement in accuracy occurs because the feedback keeps students on the solution path. If students cease to follow a solution path, either because they get stuck or they make a mistake that takes them off the path, then all the mastered rules that could apply further down the path will not have the opportunity to apply. If students cease following the path quite early, then it is almost as if they had not done the problem at all. Moreover, the more unmastered rules, the earlier the student will cease following the solution path. If Andes students know very little physics, then even if they are given many problems, they will make few correct entries. Indeed, the log files of simulated students with few mastered rules are usually quite short—they have few entries in them.

When an assessor has little evidence of success, few rules will receive credit, and that hurts its accuracy at detecting mastered rules. Only the rules that can be applied early in the problem will tend to get credit. The other rules, which tend to be applied toward the end of the solution path, will seldom have an opportunity to apply, so there will be little evidence for them in the students' behavior, so their posterior probabilities of mastery will be low, and Andes will not count them as mastered. Thus, Andes' assessment of mastered rules will be quite inaccurate. The student actually knows more rules than Andes can give it credit for, but those rules never get the chance to apply.

This is not a failure of Andes' assessor. No reasonable assessment algorithm should credit a rule with mastery when it hasn't seen evidence of it applying. If students are not kept on the solution path, then there just isn't enough data to do a good job of detecting mastered rules, so it doesn't matter what kind of assessment algorithm is used.

On the other hand, if there is some kind of immediate feedback, then whenever an unmastered rule fails to apply (or a buggy rule applies in its place), the feedback supplies a correct proposition, and thus allows mastered rules further along the solution path to apply. Rules that can only apply near the end of the solution path now have the opportunity to apply, so the assessor can finally detect them. Immediate feedback, or anything else that tends to keep students on the solution path, increases the accuracy at detecting mastered rules.

Actually, "following a solution path" isn't quite the right image, because the solutions in this task domain do not consist of a single path. Instead, a solution consists of a tree-shaped derivation of the final answer (see Figure 5; the M nodes are mastered rules, the U nodes are unmastered rules; the A nodes are rule applications, and the P nodes are propositions). Each rule application in the tree has a rule as a parent, as well as some propositions higher in the tree.

Even though Andes allows students to enter propositions in any order, students tend to start at the top and work down. Even so, there is certainly more than one solution path.

Now if any rule is unmastered, then that rule's conclusion is absent from working memory (assuming a lucky guess does not occur), so all propositions that depend on the absent proposition are also absent, including the answer. Thus, a huge section of the tree is absent just because one rule was unmastered. In Figure 5, there is only one unmastered rule, but all the shaded nodes depend upon it. Even though there are many mastered rules beneath the unmastered one, none of them can get any credit because they never have the opportunity to apply. That is, a single unmastered rule can hide many mastered rules.

On the other hand, if the tutor gives immediate feedback on all propositions, then even though the highest gray proposition is absent, the rule application just below it sees the correct proposition, so it produces a correct (non-gray) proposition because it is a mastered rule. Thus the whole chain mastered rule applications below it is present, and the only gray nodes left are at the top. Thus, immediate feedback prevents inaccurate assessment of mastered rules.

Although the solutions in this domain are tree-shaped, we will continue to refer to this assessment problem as "keeping students moving along the solution path." The problem occurs when student turn off the solution path (make a buggy or mistaken inference) onto an incorrect, garden path. It also occurs when students get stuck and stop moving down any path. The essential problem is that the students stop producing evidence that the assessor can interpret in terms of rule applications. When this occurs, any assessment algorithm will have low accuracy at detecting mastered rules simply because it receives no evidence for most of them. That is, the problem is in the impoverished data given to the assessment algorithm, and not in the assessment algorithm itself.

Increasing the number of observable nodes increases accuracy

So far, we've explained why the assessors with feedback (F and C as the prefix) are more accurate than the assessors with no feedback (N as the prefix). Why are the assessors with all nodes observable (A in the middle) more accurate than assessors with only some nodes observable (O in the middle)? Feedback only occurs on the observable nodes, so the more observable nodes, the shorter the "gray paths" in the solution tree, and the more mastered rules are exposed for detection by the assessor. In a sense, having no feedback at all is just the

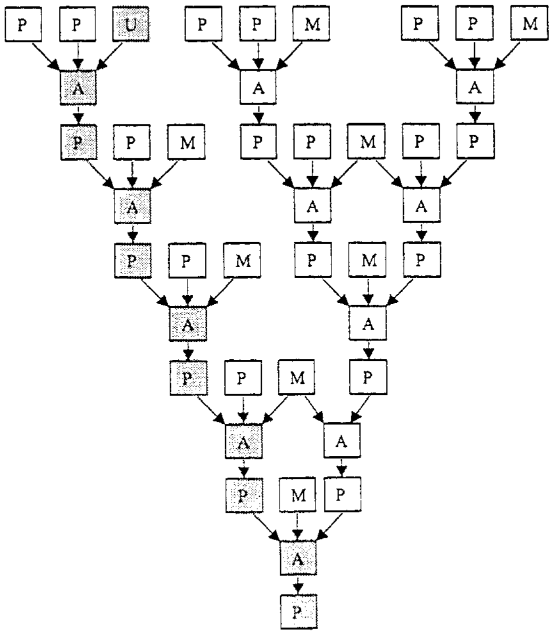


Figure 5. A Bayesian net where one unmastered rule (U) hides many mastered rules.

extreme case of unobservability. Thus, the assessors with feedback and only some observable nodes (O in the middle: FOP, FON, COP and CON) should be halfway between the no-feedback assessors (N prefix: NAP, NOP, NAN, NAP) and the feedback assessors that have all propositions observable (A in the middle: FAP, FAN, CAP and CAN). In Figure 4, the feedback assessors with only some observable nodes are somewhat lower than expected. Indeed, they are mixed together with the no-feedback assessors in Figure 4 and constitute the lower group of 8 assessors.

Upon reflection, the reasons for this finding becomes clear. The observability of a proposition depends on its type. For instance, goals are usually not observable but vectors are always observable in Andes. Some rules will have lots of conditions that are unobservable, and will always be unobservable in any problem. If any of the rules that produce the unobservable conditions are unmastered, then they will not produce the condition that our rule needs, so it will not apply. It only gets a chance to fire when all the rules feeding it unobservable conditions happen to be mastered, or just the right combination of lucky guesses occurs. Thus, it will take a long time for the rules with unobservable conditions to get an opportunity to fire, and thus they take a long time to accumulate enough credit to cross the assessor's threshold. In short, because observability is determined by the content of the rules and not just randomly distributed among the propositions, certain rules seldom get enough credit to rise above the threshold when they are mastered. Thus, the benefits of feedback are much greater when all nodes are observable than when only some nodes are observable. This explains why the feedback assessors with O in the middle (FOP, COP, FON and CON) are mixed together with the no-feedback assessors (N prefix) in Figure 4.

Although the difference between the lower 8 assessors and the upper 4 assessors is quite clear in the upper and lower panes of Figure 4, the jump appears in the wrong place in the middle pane. (It appears just before NAP instead of just after it.) That is, NAP (no feedback, all nodes observable, positive evidence only) when there is guessing has accuracy that is almost as high as the best assessors. However, because there is no feedback, few mastered rules have the opportunity to apply, so most of the propositions that are true are caused by guessing. Because all propositions are observable, whenever there is a guess, some rule gets credit for it even if it doesn't deserve it. Thus, all rules, regardless of whether they are mastered or not, tend to rise and cross the threshold. Thus, mastered accuracy is high, but unmastered accuracy is abysmal (as discussed later).

In short, there really are only two groups: the lower 8 consisting of both no-feedback assessors and assessors with only some observable nodes, and the upper 4 consisting of feedback assessors with all nodes observable. The difference is caused by immediate feedback keeping students moving along the solution path, which is most effective when all nodes are observable.

Negative evidence usually hurts accuracy at assessing mastered rules

Inside both the group of 8 and the group of 4 assessors, the assessors that allow both negative and positive evidence (N suffix) have lower accuracy at detecting mastered rules than the assessors that use positive evidence only (P suffix). The reason for this is simple. The main effect of negative evidence is to spread blame to rules. If any of the blame is spread to mastered rules, then mastered accuracy is hurt.

Moreover, this sort of damage occurs more frequently with the no-feedback assessors. Because not having feedback makes many propositions false (i.e., there are many entries that students can't make because they ceased following the solution path earlier), the no-feedback assessors that allow negative evidence (NAN and NON) spread lots of blame around, and thus are much worse than the no-feedback assessors that do not allow negative evidence (NAP and NOP).

In summary, there are basically two important effects that explain the trends in the master accuracy data. (1) Keeping students moving along the solution path via immediate feedback helps accuracy at assessing mastered rules, and making all nodes observable increases its effectiveness, and (2) Negative evidence hurts mastered accuracy, but its effects are ameliorated by immediate feedback.

ACCURACY AT ASSESSING UNMASTERED RULES

Figure 6 shows the graphs for accuracy at assessing unmastered rules. The same conventions are used as in Figure 4. Coincidentally, there are essentially just three trends responsible for their shapes:

1. In the top pane of Figure 6, most points are near 1. That is, accuracy is usually high when students do not guess.
2. In the middle pane of Figure 6, the curves rise abruptly when the assessors change from a P suffix to an N suffix. Also, when the assessors are sorted so that the average accuracy is monotonically ascending (bottom pane), the assessors with an N suffix are more accurate than assessors with a P suffix. That is, assessors that use both negative and positive evidence (N suffix) have higher accuracy than assessors that use just positive evidence (P suffix).
3. In both panes, the curves from assessors with high thresholds are higher than those from assessors with low thresholds. That is, accuracy is higher when the threshold is higher.

Subsequent sections discuss each trend.

Raising the threshold of mastery increases accuracy at detecting unmastered rules

The graphs for the 0.95 threshold were generally above those for the 0.80 threshold. This makes sense. This kind of accuracy is the proportion of unmastered rules that are below threshold, so raising the threshold improves accuracy. As mentioned earlier, raising the threshold hurts the other kind of accuracy (proportion of mastered rules that are above threshold), so there is a tradeoff.

Negative evidence raises accuracy at assessing unmastered rules

Of the features that define assessors, negative evidence raised the accuracy of assessing unmastered rules the most. The type of feedback and the proportion of observable nodes had little effect, whereas all the assessors with negative and positive evidence (N as the suffix) had higher mean accuracy than any of the assessors with positive evidence only (P as the suffix).

It is easy to see why negative evidence was the most influential feature. An unmastered rule in a simulated student usually produces a false proposition, which corresponds roughly to the student not making an entry. When the assessor uses negative evidence, false propositions are clamped in the assessor, so the unmastered rules are blamed, which drives their posteriors down, and thus raises unmastered accuracy. On the other hand, when only positive evidence is used, then there is never any blame to drive posteriors down. In fact, since there is no negative evidence, a rule's posteriors can only rise and never fall. If unmastered rules are somehow involved in producing a piece of positive evidence, they will get some credit and their posteriors will rise. If this happens enough, their posteriors will rise above the threshold and thus harm the accuracy at detecting unmastered rules. In general, adding negative evidence helps accuracy at assessing unmastered rules while using positive evidence only harms it. To put it in terms of tutor design, if you want more accuracy at detecting unmastered rules, design the user interface so that whenever an entry is not made by the student, you can safely assume that the student is *unable* to make it.

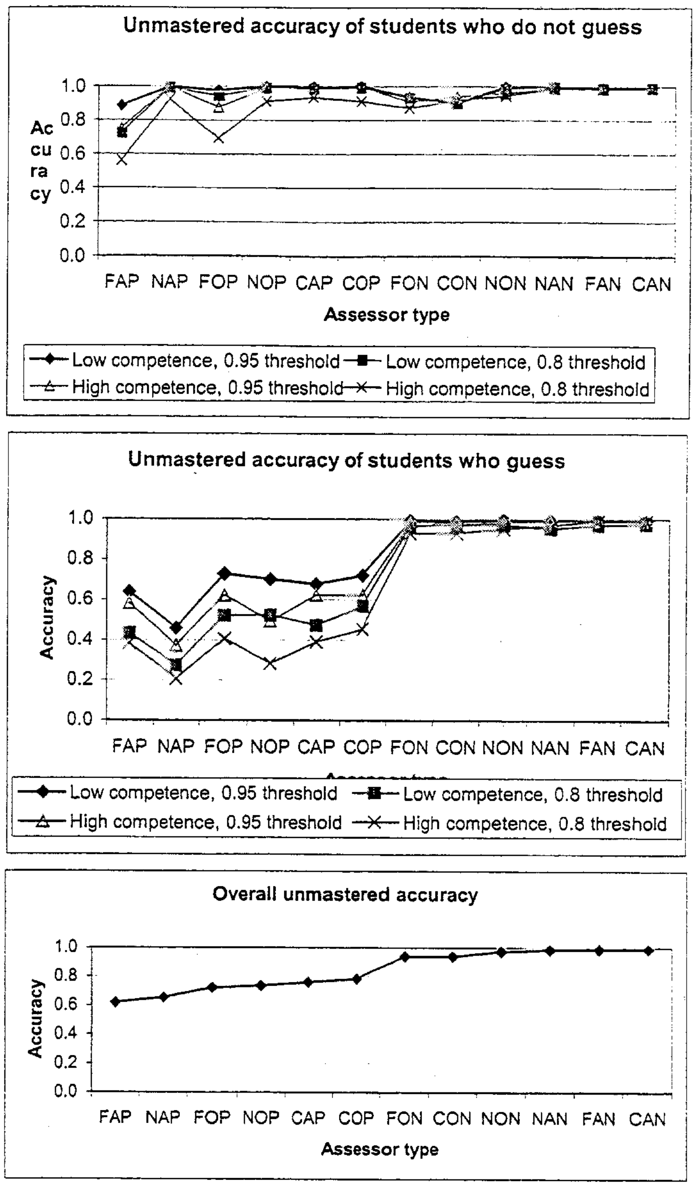


Figure 6. Unmastered accuracy

Accuracy is usually high if there is no guessing

The top graph of Figure 6 shows that when there is no guessing, accuracy at assessing unmastered rules is almost always high. This makes sense. When there is no guessing, all the positive evidence is due to application of mastered rules. Thus, the only way that an unmastered rule can get credit is when it and a mastered rule both produce a true proposition. This occurred infrequently, because there are few propositions with multiple derivations. Thus, unmastered rules rarely get credit for positive evidence, and their posterior probabilities do not rise. On the other hand, if guessing occurs, then some propositions that are produced only by unmastered rules will nonetheless be true. This causes the unmastered rules to receive some credit for mastery even though it was a lucky guesses that produced the true propositions.

There is one exception to this generalization. When flag feedback is used (F prefix) and only positive evidence is used (P suffix), then even though there is no guessing, unmastered accuracy is only mediocre (i.e., assessors FOP and especially FAP). The reason for this apparent anomaly are complex, so they won't be explained in detail. The bottom line is that mandatory correction assessors (C prefix) do not let credit flow upwards, but both flag feedback (F prefix) and no feedback (N prefix) do let credit flow upward. This would hurt the accuracy of both flag feedback and no-feedback assessors, except that the no-feedback assessors have few entries in their log files, so there is little credit to flow upward and damage accuracy. Thus, only the flag feedback assessors are hurt.

In summary, the results concerning accuracy at assessing unmastered rules are: (1) When there is no guessing, unmastered rules usually get no credit, so their posteriors stay close to their priors, and accuracy is high; only flag-feedback assessors with positive evidence are an exception. (2) When there is substantial guessing, then negative evidence is needed to counteract the undue credit that unmastered rules receive from lucky guesses.

OVERALL ACCURACY

Having discussed the accuracy at assessing mastered rules and at assessing unmastered, it is time to consider what kinds of assessors are best overall. This means finding a way to combine the two kinds of accuracy into a single measure of overall accuracy. Logically, this means defining a measure *A* such that

$$A = f(g(M), h(U))$$

where *M* is accuracy as assessing mastered rules, *U* is accuracy at assessing unmastered rules and *f*, *g* and *h* are monotonic functions. Such an *A* has the property that it rises as either accuracy rises, and there is some kind of trade off between the two types of accuracy which is represented by the functions *f*, *g* and *h*. It turns out that the standard choices for these functions are inappropriate for our purposes. We will discuss them first, then present the functions that we chose instead.

Standard measures of average accuracy

One way to assess overall accuracy is to use d-prime, which is the standard measure of sensitivity in signal detection theory (MacMillan & Creelman, 1991). D-prime uses addition for *f*, and the z-transformation for *g* and *h*. Thus,

$$\text{d-prime} = \text{z-transform}(M) + \text{z-transform}(U).$$

The z-transform (denoted by Normsinv in Excel) is graphed in Figure 7.

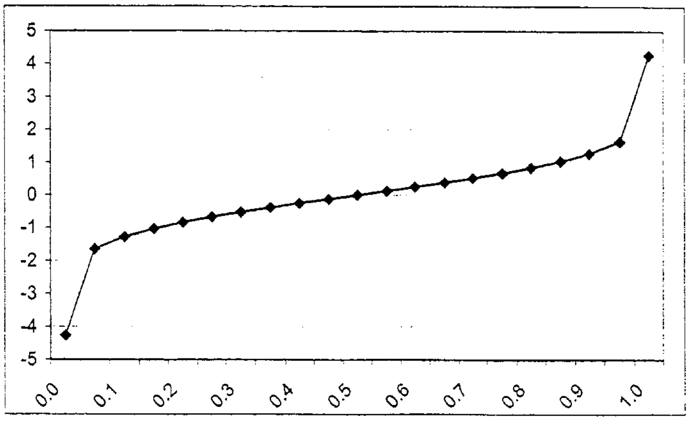


Figure 7. The z-transform

The chief advantage of d-prime is that its value is independent of the assessor's threshold, given that the posterior probability distributions of both the mastered and unmastered rules are normally distributed. Since thresholds are set rather arbitrarily, d-prime should be a useful way to measure overall accuracy. Unfortunately, the posterior probability distributions are not normally distributed. For instance, there are sometimes multiple peaks, as Figure 3 shows.

Even so, we could still use d-prime to combine mastered and unmastered accuracy even though the resulting measure now depends on the threshold.

However, as one would expect given the shape of the z-transformation curve (Figure 7), d-prime favors assessors where one or both of the accuracies is near 1.0 Hence, it favors assessor types such as NAN, where almost all rules are below threshold thus making the proportion of unmastered rules below threshold be nearly 1.0 (i.e., it thinks everything is unmastered, so it is 100% right on the unmastered rules). One would certainly not want to use such a pessimistic assessor, yet d-prime favors it. Clearly, we need a different measure than d-prime for combining mastered and unmastered accuracy.

Average accuracy as a measure of overall accuracy

As Figure 7 indicates, for most of its range, the z-transformation is close to linear. It becomes non-linear only when it is close to 1 or 0. The non-linearity of the curve close to 1 is what is causing the nonsensical ranking of assessors. This suggests simply using a linear transformation. In particular, we could simply use the average accuracy:

$$A=(M+U)/2$$

This measure uses addition for *f* and *x/2* for *g* and *h*. It assumes, as does d-prime, that overall accuracy should weigh mastered accuracy and unmastered accuracy equally. In some applications, equal weighting might not make sense. Yet we must use some weighting, so let us use a symmetric one.

Average accuracy is dependent on the choice of threshold, so our first step should be to check how the ranking of assessors changes when the threshold changes. Figure 8 graphs the average accuracy of the assessors for the 0.95 threshold and the 0.8 threshold. The x-axis indicates the assessor types, and is sorted by the mean of the two average accuracies. It is clear that varying the threshold will make only minor changes in the rankings of the assessors.
In order to evaluate the assessors, it is helpful to have a bit more data visible, and to sort the assessors in a slightly different order. Figure 9 graphs three kinds of accuracy:

- the overall accuracy on mastered rules. This is a repetition of the lower pane of Figure 4, with the x-axis ordered differently.
- the overall accuracy on unmastered rules. This is a repetition of the lower pane of Figure 6, with the x-axis ordered differently.
- the average of the preceding two accuracies.

The bottom curve shows the standard deviation over the 16 data points per assessor that are shown in Figures 4 and 6. The x-axis shows the assessors and is sorted by their average accuracy, except that NAN and FOP have been switched.

The curves suggest that there are essentially three groups of assessors. In left to right order, they are:

1. NOP, NAP, NON and NAN. These are the no-feedback assessors (N prefix).
2. FOP, COP, FON and CON. These are the feedback assessors (F or C prefix) with only some propositions observable (O in the middle).
3. FAP, CAP, FAN and CAN. These are the feedback assessors (F or C prefix) with all propositions observable (A in the middle).

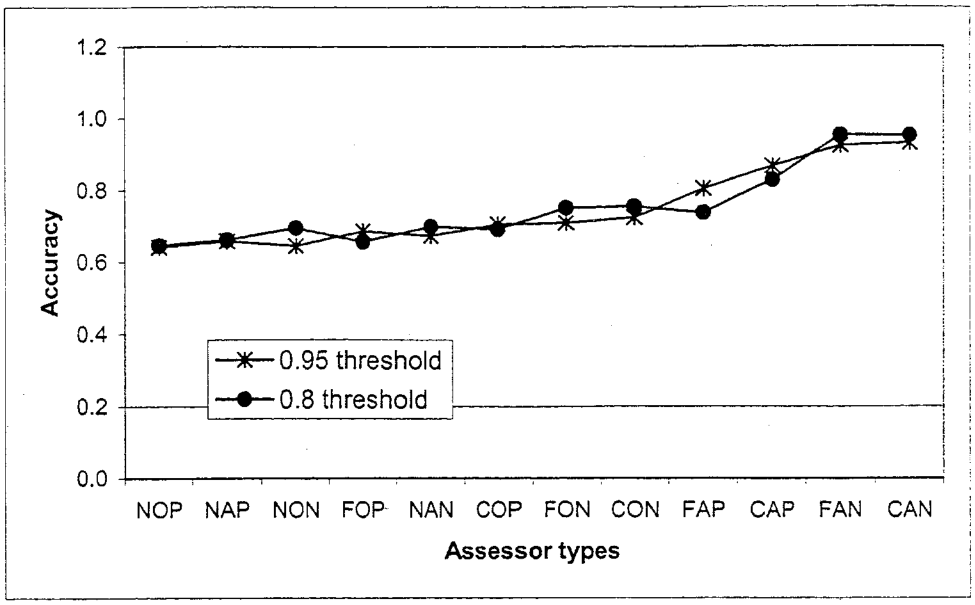


Figure 8: Average accuracy at two thresholds

Within each group of four, the two assessors that receive positive evidence only (P suffix) have clearly different behavior than the two assessors that receive both negative and positive evidence (N suffix). Thus, one can think of the assessors as 3 groups with two subgroups each. The subgroup that allows negative evidence has high unmastered accuracy and low mastered accuracy, while the subgroup with positive evidence only has moderate values for both mastered and unmastered accuracies.

The best assessors are FAN and CAN: they have some feedback (prefix of F for flag and C for mandatory correction), all nodes observable (A in the middle) and both negative and positive evidence (N suffix). To see why these two are the most accurate assessors, let us consider CAN first. No rule application depends on the occurrence of any other rule application. If a rule failed to apply in the simulated student, then the mandatory correction feedback would correct its output so that subsequent rule applications would "see" a correct proposition instead of a missing or incorrect one. Thus, for every rule, regardless of what happens to its parents, all the input propositions to the rule will be in working memory, so the rule will apply if and only if it is mastered. For assessors other than CAN, a rule might be mastered but could be prevented from applying because some rule higher in the tree failed to produce the proposition that this rule needs as input. Thus when the tutor employs mandatory correction feedback and all nodes are observable, and the assessor is given an entry corresponding to a proposition output by a

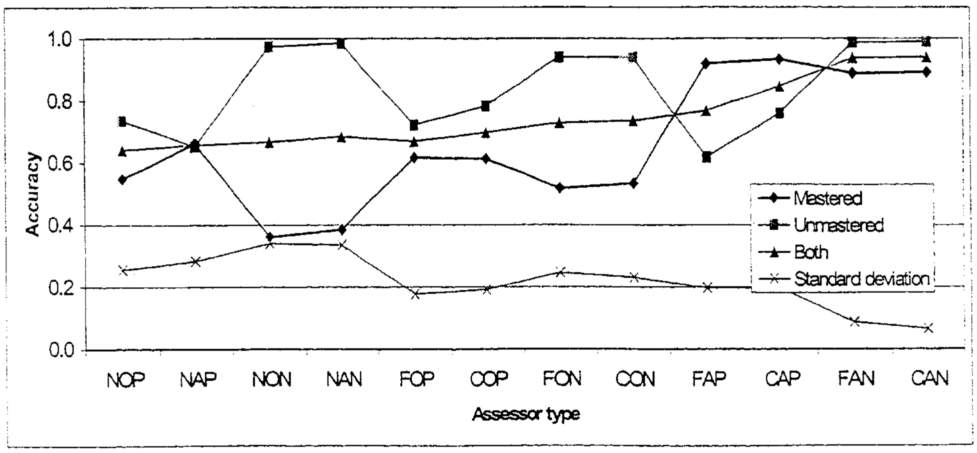


Figure 9. Average Accuracy

rule, the assessor has only to decide whether it was the rule or a guess which caused the entry. If the entry corresponding to that proposition is missing, then the assessor only has to decide whether its absence was due to a slip or a lack of mastery. It is almost as if the student were tested on each rule individually. Seen this way, it is obvious why CAN assessors are the most accurate.

If the feedback is only flag feedback (FAN), then there is only an 80% chance that feedback will correct a false parent instead of a 100% chance. Yet that evidently suffices for high accuracy, given that all propositions are observable.

In short, the FAN-CAN subgroup has high accuracy because the assessors check each rule application in near isolation from the others.

The next best subgroup consists of FAP and CAP, that is, assessors with flag (F) or mandatory correction (C) feedback, all propositions observable (A) and positive evidence only (P). For these assessors, the accuracy at assessing mastered rules is almost as high as that of FAN and CAN (same assessors, but with negative evidence allowed). However, the lack of negative evidence reduces the accuracy at detecting unmastered rules. In fact, FAP has the lowest unmastered accuracy of all the assessors.

Moreover, the standard deviation is higher for the positive-only assessors (FAP and CAP) than for the assessors that interpret absent entries as signs of ignorance (FAN and CAN). A large standard deviation means that the accuracy varies considerably depending on whether the student has high or low competence, guesses a lot or a little, etc. Although one wants the assessment of a particular student to be sensitive to their competence, guessing, etc., one does not want the *accuracy* of the assessment to be sensitive to such variables.

DISCUSSION

It is often thought that the most difficult problem facing an assessor is assignment of credit and blame. However, by doing a systematic sensitivity analysis, we have found that assignment of credit and blame was actually the least important problem for Andes. Other problems were more important. Let us discuss these problems one by one.

Keeping students moving along the solution path

The whole reason for having students solving problems during assessments is to see which of the target rules will be applied by the student. However, when solving a problem requires chains or trees of rule applications, and the student ceases following the chain or tree early in the solution, then the remaining opportunities to display rule application are wasted. If the student has mastered a correct rule, it can't be applied because the information it requires can't be inferred by the student. This is a fundamental problem with all assessors, and not just a problem with Andes and its Bayesian networks. The problem is in the impoverished data given to the assessment algorithm, rather than the assessment algorithm itself. No assessor can detect mastered rules that were denied the opportunity to apply.

Logically, there are three approaches to solving the problem:

1. Prevent students from leaving the solution path (let's just assume for simplicity that the solutions are path-like).
2. Let them leave the solution path, but keep the path so short that little data is lost.
3. Let them leave the solution path, but analyze the resulting incorrect entries.

Each technique will be discussed briefly below.

Immediate feedback: In this paper, we explored only the first solution to the problem: keeping students moving along the solution path. We measured the accuracy that resulted from using flag feedback and mandatory correction feedback to repair rule application failures. We discovered that assessors that use either form of feedback can achieve high accuracy. As the ranking of assessors in the Figure 9 shows, keeping students moving along the solution path was more important than any other parameter we varied, in that the assessors that used some kind of immediate feedback (F or C as a prefix) and had all propositions observable (A as the middle letter) were significantly better than all the others.

However, we also discovered that feedback increases accuracy only when the bandwidth (see below) is very high. If only some of the propositions can be entered on the tutor's user interface (assessors with O as the middle letter), then some rule applications are still hidden by earlier rule application failures, so accuracy is mediocre. It is only when students are required to make entries after each inference (A as middle letter) that lost opportunities to apply rules are restored and accuracy makes a significant jump.

In retrospect, it is clear why increasing bandwidth is necessary to make immediate feedback effective. Whether or not a proposition is observable depends on what it is. Goals are typically not observable, for instance. If a rule has unobservable propositions as its conditions, then it will always have them as conditions, which means that earlier mistakes can always prevent its conditions from being available. Thus, this rule's application is always confounded with other rules' applications, making it difficult to assess. Increasing the length of the test, for instance, will not help much because it is the structure of the knowledge that makes it hard to assess this rule.

Short solution paths: The second solution listed above is to minimize the damage done by using short solution paths. Thus, if a student leaves the path early and the resulting entries cannot be used for assessment, only a few entries are wasted. A tutor or assessor built around this solution to the problem would resemble current standardized tests and CAI tutors in that the problems would take students only a short time to solve. Of course, posing only short problems prevents assessment of reasoning that can occur only in more complex contexts (Collins, 1990; Linn et al., 1991).

Analysis of entries based on errors: Perhaps the most promising solution to the problem is to enhance the power of the assessor so that it can analyze entries that depend on errors made earlier. This would allow it to detect applications of correct rules even when the applications are based on incorrect input propositions. There are several methods for achieving this kind of power in the assessor. One is to equip the system with so many incorrect (buggy) rules that it can anticipate almost any error that the student is likely to make. Buggy rules can even be used to represent unstable errors, which might be more accurately represented as incorrect applications of correct knowledge (VanLehn, 1990). Using buggy rules means that the system can find a derivation (i.e., a tree of rule applications such as the one in Figure 5) leading up to an incorrect entry just as easily as it can find one leading up to a correct entry. The derivation of the incorrect entry will contain applications of both incorrect (buggy) rules and correct rules. This means that correct rules used in deriving the incorrect entry still get the credit that they deserve. They will not be hidden by application of buggy rules. Unfortunately, this solution to the problem requires collecting an extensive library of bugs, and that is notoriously difficult (VanLehn, 1988).

Another solution to the problem is to let the student make incorrect entries, then accept them as new information, akin to the given information that describes the original problem. In this design, the problem solver has only correct rules so it can not analyze the incorrect entries themselves (the design that uses bugs can analyze incorrect entries). However, it can at least explain subsequent entries that are based on the incorrect one. For instance, suppose the student incorrectly enters "$g=-9.8$" (an incorrect physics equation) then applies the correct rule "$W=m*g$" and enters "$W=-9.8*m$." The solver cannot figure out how the student derived $g=-9.8$. However, it can take $g=-9.8$ as new information and derive $W=-9.8*m$ using its correct $W=m*g$ rule, which allows it to give the student credit for knowing that rule. Unfortunately, the problem with this design is that the student can get the problem solver into self-contradictory states (e.g., two values for the same variable), or states from which it is impossible to reach a goal. Perhaps because of these technical difficulties, no ITS that we know of has tried this solution.

Yet another solution to the problem is to cease trying to recognize whole lines of reasoning and only recognize isolated applications of specific pieces of knowledge. This ITS architecture was called *issue recognition* by the first ITS to use it, West (Burton, 1982). More recent examples of this architecture, such as SQL-Tutor (Mitrovic & Ohlsson, 1999), Smithtown (Shute & Glaser, 1990) and TRIO (Ritter & Feurzeig, 1988), represent domain knowledge as constraints, demons or critics. Each has an applicability condition and a correct action. When the applicability condition is met, even in a state that is derived from earlier incorrect actions, the constraint (or issue, demon, etc.) receives credit if the student's action is the same as the constraint's. The constraint receives blame if the student's action is incompatible with the constraint's. This solves the problem of students leaving the solution path because mistakes made earlier do not prevent recognition of subsequent correct reasoning. On the other hand, if the student is in a state that does not happen to match any constraint's applicability condition, then the student's action in that state has no effect on the assessment. If this occurs often, then once again we have too little data on the student's performance. For instance, Smithtown's constraints seem to have been rarely applicable, so it "was more discovery learning than guided" (Shute & Glaser, 1990, pg. 74) and thus had too little data to form an accurate student model.

All these techniques (buggy rules, using incorrect information as given, or issue recognition) are based on following the student as the student leaves the solution path, so they will not help when the student simply gets stuck. The other techniques—immediate feedback and using short-answer problems—help reduce data lossage due to both veering off the solution path and stopping prematurely.

In short, there are many solutions to the problem of losing potential evidence of mastery—feedback, short problems, buggy rules and issue recognition. However, they all have their own drawbacks, and some have important interactions with the pedagogical policies of the tutor. This problem is important, unsolved, and deserving of further research.

Bandwidth

VanLehn (1988) defined the bandwidth of a tutor to be the degree of match between the entries that a student could make on the user interface and the inferences made by the student model. If every inference by the model corresponded to some entry on the user interface, then the tutor was said to possess a high or "mental states" bandwidth, because the states of the user interface exactly matched the states of the student model, and the student model's states approximate the student's mental states. On the other hand, if only some of the model's states corresponded to user interface states, then the tutor was said to possess a "intermediate states" bandwidth. Lastly, if only the final answer of the problem was available for the tutor to assess, as in PROUST (Johnson, 1990), then the tutor was said to be characterized by a low or "final states" bandwidth.

In this sensitivity analysis, we found that bandwidth was indeed important, albeit not quite as important as keeping the student moving along the solution path. (The 1988 review of student modeling did not mention the problem of keeping students moving along the solution path.) We compared assessors with all nodes observable (A as the middle letter in the name) and with only 80% of the nodes observable (O as the middle letter). That is, we compared a mental states bandwidth to an intermediate states bandwidth. The feedback assessors (F and C as the prefix) were significantly affected by this bandwidth. As discussed earlier and displayed in Figure 9, making even just 20% of the nodes unobservable reduced the assessors ability to identify mastered rules from about 90% accuracy to 50% accuracy. In short, tutors that need a high accuracy assessment should use as high a bandwidth as possible.

However, there is a second aspect of the correspondence between user interfaces and the cognitive model that was also not mentioned in the 1988 review and yet is quite important. It concerns the difference between tutors that require students to show all their work (assessors with an N suffix) and tutors allow students to omit entries if they want (assessors with a P suffix).

Forcing students to show their work: The Andes user interface is intended to be as much like a piece of paper as possible. This should both facilitate learning the user interface and increase transfer from the tutor to unsupported, paper and pencil problem solving. A seemingly minor point in the user interface design is that students are not forced to enter any specific vectors, formulas or equations. They can enter any ones they want. In contrast, other physics tutors (e.g., Reif & Scott, 1999) prompt students for entries (e.g., "Draw the weight vector." or "What is an expression for the magnitude of W?"). Such tutors will not go on until the requested entry is provided. If the student leaves the entry blank or fills it incorrectly, then the tutor can infer that the student *can't* make that entry correctly. If the tutor were using a Bayesian assessor, it could clamp the appropriate nodes to False, indicating that the student cannot perform that action. Thus, such a tutor could use negative evidence. However, in Andes, the fact that a student has not entered a certain equation does not imply that the student *can't* enter it. The student may enter it later, or may just hold it in working memory instead of entering it on the Andes user interface. For instance, many students prefer not to draw vectors but instead start immediately to enter equations. Andes cannot assume that they don't know how to draw the missing vectors, for they may be envisioning them mentally even though they have not drawn them. Thus, Andes can only clamp nodes when it observes the corresponding entries, and thus it only clamps nodes to True. Thus, Andes can only use positive evidence (nodes clamped True) and not negative evidence (nodes clamped False). This is not just a peculiar feature of Andes. In general, tutors that force student to enter specific information can use both negative and positive evidence, while tutors that let student's enter whatever they want can only use positive evidence.

We discovered that giving students the freedom to omit entries harms the assessment. As Figure 9 shows, when the assessor can use only positive evidence (P suffix), then its ability to identify unmastered rules decays precipitously, thus pulling down the overall accuracy of the assessor. Moreover, as Figure 4 shows, as students guess more frequently, the positive-evidence-only convention hurts accuracy even more. Moreover, leaving out negative evidence harmed all assessors, regardless of the values of their other structural or numerical parameters.

Using positive evidence only should reverse our normal intuitions about test length. Although one usually thinks that giving a longer test will increase the accuracy of the assessment, if only positive evidence is used, then *longer tests will actually lower accuracy.* When there is no negative evidence to drive a rule's probability of mastery down, then it can only rise. Even if lucky guesses occur infrequently, given enough problems, any unmastered rule's posterior will eventually exceed the threshold. Thus, longer tests harm the tutors ability to identify unmastered rules, thus reducing the overall accuracy.

Our conclusion is that increased accuracy requires that the tutor somehow require that students "show all their work." This allows absent work to be interpreted as an error of omission, and negative evidence can thus be entered into the assessor. This amounts to telling students that they must make all relevant entries, because if they don't, the tutor will assume that they can't. For instance, in some tutors, student must fill in all the blanks that the tutor presents. Unfortunately, such heavy scaffolding may reduce transfer to naturally occurring "user interfaces" such as pencil and paper.

One possible solution is to allow students the freedom to omit entries as long as all the entries they make are correct. As soon as they make an incorrect entry, they are required to show all the work leading up to that entry. During this show-work phase, omitted entries are clamped False, thus providing the negative evidence that is needed for accurate assessment.

Assignment of credit

When this research began, we thought that the most important problem for the Andes assessor was the classic assignment of credit problem: if a proposition has multiple derivations and it is true, how much credit should be given to each derivation? The Bayesian approach is to apportion the credit according to the prior probabilities of the derivations, so derivations with higher priors get more credit. This "rich get richer" approach seems unfair, but it is mathematically sound. It would have been interesting to see if it yielded high quality assessments.

However, it turned out that this particular assignment of credit problem rarely occurs in our task domain. Less than 3% of the propositions had multiple derivations. Removing them hardly increased the accuracy at all.

However, there is a second assignment of credit problem, which is often overlooked. The assessor must decide whether a proposition is true because the student guessed correctly or because the student applied a mastered rule. That is, rules and guesses compete for credit for correct entries. This competition did play a role in explaining many of the results.

Assignment of blame

When negative evidence is introduced, an assessor must face the classic assignment of blame problems. Just as there are two assignment of credit problems (apportioning credit between a rule and a guess; apportioning credit between multiple rules), there are two assignment of blame problems:

1. If an expected correct entry did not occur, did a slip occur or not? By definition, a slip occurs when all the propositional parents of a rule application are in working memory and the rule is mastered, but the rule application still does not occur.
2. If an expected correct entry did not occur and a slip does not occur, then is the rule unmastered or are some of the propositional parents absent from working memory, and if so, which ones?

Andes and the assessors tested here use a slip parameter that is so low (0.001) that slips absorbs little blame. The slip parameter is essentially part of the definition of mastery. That is, by setting the slip parameter to 0.001, we are saying that students have mastered a rule if they fail to apply it at most once in a thousand occasions where they should apply it. This is perhaps an overly stringent definition of mastery, and we should have experimented with different values.

However, we now favor a more accurate model of competence. The basic idea is to dispense with the binary model of mastery entirely and use a continuous one instead. In particular, mastery should be defined as the probability that a rule is applied given that all its propositional parents are true. A new assessor is being implemented with this model of mastery, and we hope to report on its accuracy in the near future.

Recommendations for tutor developers

The most accurate assessors in the study (see Figure 9) were FAN and CAN. They used immediate feedback (F for flag feedback; C for mandatory correction feedback), required that all nodes be observable (A in the middle) and required students to show all their work (N suffice). However, high accuracy assessment might compromise the pedagogical policies of the tutor.

The issue of the benefits of immediate feedback have been investigated (Anderson et al., 1995; Kulik & Kulik, 1988; Mark & Greer, 1995). It appears that immediate feedback speeds up learning without harming the eventual understanding.

The pedagogical effects of bandwidth seems to require more study. Although FAN and CAN have high accuracy, they require that all actions be observable and none can be skipped. This means that the grain size of the user interface must be small and/or the grain size of the knowledge representation must be large. It is a necessary consequence of trying to enforce a one-to-one relationship between student inferences and actions. Such a one-to-one correspondence has a price. Because people can think much faster than they can operate a user interface, a large-grained knowledge representation may be a gross approximation. A small grain-size user interface will probably be tedious to use. Most importantly, a small grain-size user interface may be quite unlike those that occur naturally (e.g., a piece of paper), which may make it difficult for students to transfer their skill from the tutor to a naturally occurring user interface.

Nonetheless, FAN and CAN do seem to work well in at least some task domains. An example of an CAN tutor is the Anderson et al. (Anderson et al., 1995) Lisp tutor. If students make an incorrect entry, they are given increasingly specific feedback until they correct the entry, so it is a type "C" assessor. For each user interface action (entering a Lisp atom or a parenthesis), there is usually just one production rule firing in the ideal student model. Thus, it is a type "A" assessor. Students may not skip actions, so it is a type "N" assessor. The assessor appears to be highly accurate, as it predicts post-test scores well. However, it often predicts that student will score slightly higher on the post-tests than they do, which could be a sign of lack of transfer (Corbett & Bhatnagar, 1997). Although the Lisp tutor is normally used with mandatory correction feedback, it can also be used with flag feedback, which makes it a FAN tutor (Anderson, Boyle, Corbett, & Lewis, 1990). Because the Lisp tutor is known to cause impressive learning (Anderson et al., 1990), it illustrates that FAN and CAN tutors are not only feasible, but can sometimes be desirable as well, at least in some task domains.

Andes is a FOP tutor: It uses flag feedback (F), has only some nodes observable (O) and permits students to omit entries if they want (P). FOP tutors have mediocre accuracy (see Figure 9), which suggests that we should not put much confidence in Andes' assessments of students' mastery. Fortunately, Andes' assessments have not been used for high stakes decision making (e.g., deciding whether students pass or fail a course) or even medium stakes decision making (e.g., mastery learning: deciding to give a student another problem that exercises certain rules because those rules are not yet mastered). Andes uses its assessments mostly to decide what hint to give when the student asks for one. In particular, if Andes sees several possible continuations, it uses the assessment to decide which one to suggest to the student (Gertner et al., 1998). This is a low stakes decision, because the student still gets a hint even if it is not a perfect hint. Clearly, if we want to use Andes' assessor for higher stakes decision making, then we need to improve it, and this sensitivity analysis has suggested several ways to do that.

Currently, our favorite option is to modify the Andes user interface so that it will change over the course of training. Early on, Andes would require an entry after each inference, and thus help the tutor maintain an accuracy model of the student (i.e., it would be a FAN tutor). As the student progresses, this scaffolding fades, and the student is allowed to do more and more work mentally, without entering it on the user interface. This would decrease the accuracy of the assessor (it would become a FOP tutor again) but would increase transfer to naturally occurring user interfaces (e.g., paper). A new version of Andes is being built that fades its scaffolding in this manner.

Recommendations for test developers

Although the analysis was conducted in the context of the student modeler of an ITS, its conclusions apply to any performance assessment. Performance assessments consider not just the final product of a students' work, but their behavior as they work (Linn et al., 1991). A typical performance assessment involves a student or a small group of students doing a project that takes several hours or days to complete. Performance assessments are becoming increasingly popular, particularly in the United States, where there is increasing political pressure to make schools and teachers accountable for their students' successes and failures. Performance assessments are seen as a "test worth teaching to" (Collins, 1990).

These results suggest that students should receive immediate feedback as they work, and that their score should be proportional to the number of times they receive feedback. If they do not receive feedback, then they may wander out of the intended solution space, which will make it impossible (or at least unlikely) that they will demonstrate competence in the target concepts and skills. For instance, suppose a group of students is given an algebra project that should take several classes to complete but they make some incorrect simplifying assumptions at the very beginning that turn it into an arithmetic project. Unless they are given early feedback to correct their assumptions, their performance data will be useless for assessing their understanding of algebra. Other findings in this analysis also have implications for performance assessments—student should show all their work (type A assessors are more accurate than type O), and a failure to show work should be interpretable as an inability to produce it (type N assessors are more accurate than type P). Considerable ingenuity will be required to create performance assessments that are still "a test worth teaching to" and yet have high accuracy.

Acknowledgements

This research was supported by the Cognitive Sciences Division of ONR under grant N00014-96-1-0260 and by ARPA/CAETI under grant number N66001-95-C-8637. We would like to thank the members of the Andes group for all their contributions to the system and for their comments on the manuscript.

References

Albacete, P. L. (1999). *An Intelligent Tutoring System for Teaching Fundamental Physics Concepts.* University of Pittsburgh, Pittsburgh, PA.

Albacete, P. L., & VanLehn, K. (2000). The Conceptual Helper: An intelligent tutoring system for teaching fundamental physics concepts. In G. Gauthier & C. Frasson & K. VanLehn (Eds.), *Intelligent Tutoring Systems: 5th International Conference, ITS 2000* (pp. 564-573). Berlin: Springer.

Albacete, P. L., & VanLehn, K. (2000). Evaluation the effectiveness of a cognitive tutor for fundamental physics concepts. In L. R. Gleitman & A. K. Joshi (Eds.), *Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society* (pp. 25-30). Mahwah, NJ: Erlbaum.

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence, 42*, 7-49.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences, 4*(2), 167-207.

Burton, R. B. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*. London: Academic Press.

Collins, A. (1990). Reformulating testing to measure learning and thinking. In N. Frederiksen & R. Glaser & A. Lesgold & M. G. Shafto (Eds.), *Diagnostic Monitoring of Skill and Knowledge Acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. (1997). On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson & C. Paris & C. Tasso (Eds.), *User Modeling: Proceedings of the Sixth International conference, UM97.* New York: Spring Wien.

Conati, C., & VanLehn, K. (2000). Further results from the evaluation of an intelligent computer tutor to coach self-explanation. In G. Gauthier & C. Frasson & K. VanLehn (Eds.), *Intelligent Tutoring Systems: 5th International Conference, ITS 2000* (pp. 304-313). Berlin: Springer.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*, 253-278.

Corbett, A. T., & Bhatnagar, A. (1997). Student modeling in the ACT programming tutor: Adjusting a procedural learning model with declarative knowledge, *Proceedings of the Sixth International Conference on User Modeling.*

Fung, R., & Chang, K. C. (1989). Weighting and integrating evidence for stochastic simulation in Bayesian networks, *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence.* Los Altos, CA: Morgan Kaufman.

Gertner, A., Conati, C., & VanLehn, K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model., *Proceedings of the 15th national Conference on Artificial Intelligence.*

Gertner, A. S. (1998). Providing feedback to equation entries in an intelligent tutoring system for Physics. In B. P. Goettl & H. M. Halff & C. L. Redfield & V. J. Shute (Eds.), *Intelligent Tutoring Systems: 4th International Conference* (pp. 254-263). New York: Springer.

Gertner, A. S., & VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In G. Gautheier & C. Frasson & K. VanLehn (Eds.), *Intelligent Tutoring Systems: 5th international Conference, ITS 2000* (pp. 133-142). New York: Springer.

Gitomer, D. H., Steinberg, L. S., & Mislevy, R. J. (1995). Diagnostic assessment of trouble-shooting skill in an intelligent tutoring system. In P. N. a. S. C. a. R. Brennan (Ed.), *Cognitively diagnostic assessment.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Johnson, W. L. (1990). Understanding and debugging novice programs. *Artificial Intelligence, 42*, 51-97.

Kulik, J. A., & Kulik, C.-L. C. (1988). Timing of feedback and verbal learning. *Review of Educational Research, 58*, 79-97.

Linn, R. L., Baker, E. L., & Dunbar, S. B. (1991). Complex, performance-based assessment: Expectations and validation criteria. *Educational Researcher, 20*(8), 15-21.

MacMillan, N. A., & Creelman, C. D. (1991). *Detection Theory: A User's Guide.* Cambridge, UK: Cambridge University Press.

Mark, M. A., & Greer, J. E. (1995). The VCR tutor: Effective instruction for device operation. *The Journal of the Learning Sciences, 4*(2), 209-246.

Martin, J., & VanLehn, K. (1993). OLAE: Progress toward a multi-activity, Bayesian student modeller. In S. P. Brna & S. Ohlsson & H. Pain (Eds.), *Artificial Intelligence in Education, 1993: Proceedings of AI-ED 93*. Charlottesville, VA: Association for the Advancement of Computing in Education.

Martin, J., & VanLehn, K. (1994). Discrete factor analysis: Learning hidden variables in Bayesian networks. Technical report: LRDC, University of Pittsburgh.

Martin, J., & VanLehn, K. (1995a). A Bayesian approach to cognitive assessment. In P. Nichols & S. Chipman & S. Brennan (Eds.), *Cognitively diagnostic assessment*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Martin, J., & VanLehn, K. (1995b). Student assessment using Bayesian nets. *International Journal of Human-Computer Studies, 42*, 575-591.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence and Education, 10*, in press.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan-Kaufmann.

Person, N. K., Kreuz, R. J., Zwaan, R., & Graesser, A. C. (1995). Pragmatics and pedagogy: Conversational rules and politeness strategies may inhibit effective tutoring. *Cognition and Instruction, 13*(2), 161-188.

Reif, F., & Scott, L. A. (1999). Teaching scientific thinking skills: Students and computers coaching each other. *American Journal of Physics, 67*(9), 819-831.

Ritter, F., & Feurzeig, W. (1988). Teaching real-time tactical thinking. In J. Psotka & L. D. Massey & S. A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned* (pp. 285-301). Hillsdale, NJ: Erlbaum.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Los Altos, CA: Morgan-Kaufman.

Schulze, K. G., Shelby, R. H., Treacy, D. J., & Wintersgill, M. C. (2000). Andes: A coached learning environment for classical Newtonian physics, *Proceedings of the 11th International conference on College Teaching and Learning*. Jacksonville, FL.

Shachter, R., & Peot, M. (1989). Simulation approaches to general probabilistic inference on belief networks, *Proceedings of the Fifth Workshop on Uncertainty in AI* (pp. 311-318). Mountain View, CA: Association for Uncertainty in AI.

Shelby, R. N., Schulze, K. G., Treacy, D. J., Wintersgill, M. C., & Vanlehn, K. (in prep.). The Andes Intelligent Tutor: an Evaluation.

Shute, V. J. (1995). SMART: Student Modeling approach for responsive tutoring. *User modeling and user-adapted instruction, 5*, 1-44.

Shute, V. J., & Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world. *Interactive Learning Environments, 1*, 51-76.

VanLehn, K. (1988). Student modeling. In M. Polson & J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems* (pp. 55-78). Hillsdale, NJ: Lawrence Erlbaum Associates.

VanLehn, K. (1990). *Mind Bugs: The Origins of Procedural Misconceptions*. Cambridge, MA: MIT Press.

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson & G. Gauthier & A. Lesgold (Eds.), *ITS96: Proceeding of the Third International conference on Intelligent Tutoring Systems*. New York: Springer-Verlag.

VanLehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., C. P. Rose, Schulze, K., Shelby, R., Treacy, D., Weinstein, A., & Wintersgill, M. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. In G. Gauthier & C. Frasson & K. Vanlehn (Eds.), *Intelligent Tutoring Systems: 5th International Conference, ITS 2000* (pp. 474-483). Berlin: Springer.

VanLehn, K., & Martin, J. (1998). Evaluation of an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education, 8*(2).

VanLehn, K., Niu, Z., Siler, S., & Gertner, A. S. (1998). Student modeling from conventional test data: A Bayesian approach without priors. In B. P. Goettle & H. M. Halff & C. L. Redfield & V. J. Shute (Eds.), *Intelligent Tutoring Systems: 4th International Conference, ITS98* (pp. 434-443). Berlin: Springer Verlag.

VanLehn, K., Ohlsson, S., & Nason, R. (1994). Applications of Simulated Students: An Exploration. *Journal of Artificial Intelligence in Education, 5*(2), 135-175.