Using Bayesian Networks to Manage Uncertainty in Student Modeling

CRISTINA CONATI,
Department of Computer Science, University of British Columbia, Vancouver, BC, V6T1Z4, Canada. conati@cs.ubc.ca

ABIGAIL GERTNER
The MITRE Corporation, 202 Burlington Road, Bedford, MA, 01730, USA. gertner@mitre.org

KURT VANLEHN
Department of Computer Science and Learning and Research Development Center, University of Pittsburgh, Pittsburgh, PA, 15260, USA. vanlehn@cs.pitt.edu

Abstract When a tutoring system aims to provide students with interactive help, it needs to know what knowledge the student has and what goals the student is currently trying to achieve. That is, it must do both assessment and plan recognition. These modeling tasks involve a high level of uncertainty when students are allowed to follow various lines of reasoning and are not required to show all their reasoning explicitly. We use Bayesian networks as a comprehensive, sound formalism to handle this uncertainty. Using Bayesian networks, we have devised the probabilistic student models for Andes, a tutoring system for Newtonian physics whose philosophy is to maximize student initiative and freedom during the pedagogical interaction. Andes' models provide long-term knowledge assessment, plan recognition, and prediction of students' actions during problem solving, as well as assessment of student's knowledge and understanding as students read and explain worked out examples. In this paper, we describe the basic mechanisms that allow Andes' student models to soundly perform assessment and plan recognition, as well as the Bayesian network solutions to issues that arose in scaling up the model to a full-scale, field evaluated application. We also summarize the results of several evaluations of Andes which provide evidence on the accuracy of its student models.

1 Introduction

One of the key elements that distinguishes Intelligent Tutoring Systems (ITS) from more traditional educational systems is their ability to interpret student actions to maintain a model of student reasoning and learning – the *student model* (Shute and Psotka, 1996). Like user models for non-ITS software, the student model allows an ITS to adapt the interaction to its user's specific needs. However, unlike non-ITS software, the ultimate goal of an ITS's interventions is to help its users learn a target instructional domain. This adds a great deal of uncertainty to the user modeling problem, because it entails inferring from the student's actions the student's degree of mastery of the target domain knowledge, a process known as *assessment* or *knowledge tracing* (Anderson, Corbett, Koedinger and Pelletier, 1995). The uncertainty regarding the student's domain knowledge affects the second kind of modeling common to many adaptive systems that aim to provide interactive help: *plan recognition*, or understanding what the user is trying to do. The uncertainty in assessment influences plan recognition because the student's knowledge of the underlying plans affects which goals are most likely to be the focus of the user's attention.

In this paper, we describe how we use Bayesian networks (Pearl, 1988) as a unifying framework to manage the uncertainty in student modeling. The models that we present have been implemented in Andes, an intelligent tutoring system designed to help students learn Newtonian

mechanics at the university level. Beside providing tailored support during problem solving (VanLehn, 1996; Gertner and VanLehn, 2000; Schulze, Shelby, Treacy, Wintersgill, VanLehn and Gertner, 2000), Andes also helps students study examples effectively, a novel task that adds new sources of uncertainty to the student modeling problem (Conati and VanLehn, 2000).

A well-known, fundamental problem of both plan recognition and assessment is the assignment of credit problem. If there are multiple explanations for a student's action, which is the most likely? That is, how much credit should be given to each explanation? This problem is exacerbated if a system does not require students to explicitly express all the steps in their reasoning, so that the actions that the system sees can be the result of a chain of inferences rather than of a single inference. Early solutions to this problem were based on heuristics or on maximizing coverage of positive evidence while minimizing coverage of negative evidence (Reggia and D'Autrechy, 1990; Polk, VanLehn and Kalp, 1995). Recognizing the limitations of these approaches, other tutoring systems opted to reduce the number of cases where multiple explanations of student actions could occur. This is done by constraining students to follow a predefined line of reasoning and requiring them to make all of their reasoning explicit (Anderson et al., 1995). However, this kind of limitation of student initiative and freedom is not an option for Andes. Andes was designed in collaboration with physics professors at the US Naval Academy, who felt strongly that all possible correct solutions should be acceptable to Andes, and that their students would simply not use a system with an overly constraining interface.

One of the main contributions of Andes' student model is that it provides a comprehensive solution to the assignment of credit problem for both knowledge tracing and plan recognition, without reducing student initiative. Andes' solution scales up an approach first introduced in (Conati and VanLehn, 1996b). Because this approach uses the same Bayesian network for solving both problems, the solutions work together. Plan recognition influences assessment while at the same time assessment influences plan recognition, and these mutual influences are mathematically sound. This is crucial for accurate student modeling in Andes, because students often do not have knowledge of all the available plans. Thus, the assessment of what a student knows should influence the probability of what solution the student is following, and vice versa. Although knowledge assessment and plan recognition are thus strongly related, very little research in plan recognition has leveraged this connection (Jameson, 1996; Carberrry, 2001).

A second contribution of Andes' student model is that it supports *prediction* of student actions during problem solving, in addition to knowledge tracing and plan recognition. Because in physics each solution can be implemented through different action orderings, being able to predict what inferences and actions a student can perform in the context of a particular solution greatly improves the help that the system can provide when the student does not know how to proceed. While several systems have used probabilistic methods to perform one or two of knowledge tracing, plan recognition and action prediction (Jameson, 1996), Andes student model has been the first to support all these modeling tasks at once.

Finally, deploying a system that supports example studying in addition to problem solving, and that covers a full-semester physics course, required finding solutions within the Bayesian network framework to several other student modeling problems. In the rest of the paper, we will show how the Bayesian network approach allowed us to devise fairly simple solutions to these problems. However, it should be stressed that Bayesian networks are just a notation, albeit a very powerful one. The real solutions to the student modeling problems we faced are certain cognitive and pedagogical hypotheses, as we will discuss. They are represented in the Bayesian

network, but that only gives them precision and not validity. The merit of these hypotheses can only be established through extensive empirical testing and refinement. The evaluations that we describe at the end of the paper are a start in this direction.

The paper is organized as follows. First the user interfaces of Andes are presented, followed by a brief description of the modeling problems that will be discussed. After this introduction, the main part of the paper describes the Andes Bayesian networks and how they are used for tutoring. The final part discusses evaluations of Andes, related research, some unsolved problems and proposed future work.

1.1 THE TASK DOMAIN AND THE STYLE OF TUTORING

The Andes system is intended to help students learn Newtonian mechanics as it is taught in introductory physics courses at the university level. Although such courses have several instructional objectives, a common one is that students learn how to solve physics problems that involve algebraic analyses of physical situations. For instance, a simple problem is

"A car starts moving with constant acceleration a at time T_0 and after 30 seconds has moved D meters and reached a velocity V_1 of 30Km/h. What is the acceleration of the car?"

This problem requires knowing what "acceleration" and "velocity" are, and how they are related. These are the key physics concepts involved. It also requires knowing that the method to solve this problem involves the equations describing motion with constant acceleration, which in turn requires knowing the distance traveled in a given time interval and the initial and final velocity in that interval. However, the students must also know some mathematical principles – for instance how to operate with vectors. Thus, the knowledge that students must master to solve such problems is a mixture of physics principles, mathematical principles and methods for applying them. In the knowledge base that represents Andes' model of the task domain, condition-action rules are used to represent all of these different kinds of knowledge. Thus, we use "rule" to refer to any of the relevant pieces of knowledge, regardless of whether the content is procedural or conceptual, physical or mathematical.

Andes is not designed to teach a physics course all by itself. Students must still attend all class meetings and read their textbook. Andes' role is to assist learners with two homework activities: studying solutions of correctly solved problems (examples), and solving problems. These two activities often consume the largest portion of a student's study time, so if Andes can increase students' learning during problem solving and example studying, then it should have a major positive impact on their overall learning in the course.

Because students have seen problems being solved in class, they are not completely naive when they start working with Andes. Thus, its pedagogical policy is to let students do as much of the work as possible on their own. Andes offers help only when students ask or when it sees a particularly compelling reason to offer unsolicited help.

1.2 ANDES' PROBLEM SOLVING INTERFACE

The Andes interface for problem solving consists of two main entry panes (Figure 1) in which students can draw vector diagrams (upper left) and enter equations (lower right), as well as the variable definitions pane, located above the equation pane, and the hint window, below the diagram pane on the left.

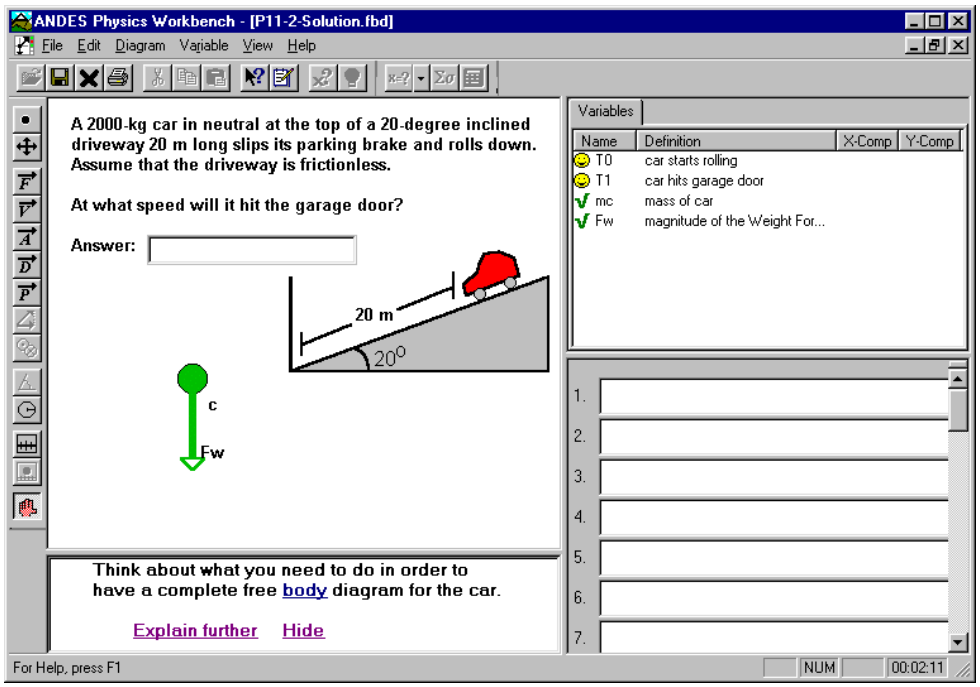

Figure 1: Andes problem solving interface

Andes provides two kinds of pedagogical interventions during problem solving: immediate feedback and help. Every entry the student makes in the problem solving interface, both in the diagram pane and the equation pane, receives immediate feedback by changing the color of the entry: green for correct and red for incorrect.

The two main types of help that Andes provides are:

- Error help. Students can select an incorrect (red) entry and can click on a button labeled "what's wrong with that?" In response, Andes generates hints suggesting what to focus on in order to fix the error.
- Procedural help. At any time, the students can ask for a hint on what would be a good next step to take.

These types of help are designed to help the student learn. This means that they often do not directly answer a student's help request. Instead, they give a hint sequence designed to encourage the student to work out as much of the answer on her own as she can. The first hint often mentions only a goal or feature involved of the correct step. Subsequent hints become more specific. The final hint in the sequence, called the "bottom-out hint" (Koedinger and Anderson, 1993), indicates exactly what step the student should enter.

1.3 ANDES' EXAMPLE STUDYING INTERFACE

In addition to the problem solving interface, Andes provides students with an interface to study examples under the supervision of the SE-Coach (Conati and VanLehn, 2000). The main pedagogical goal of the SE-Coach is to get students to self-explain examples, that is to clarify to themselves why and how each line of an example solution is derived. The SE-Coach is specifically designed to counteract the tendency that many students have to not self-explain, a tendency that can hinder effective learning from examples (Chi, in press). Thus, it needs to

monitor whether students self-explain spontaneously and encourage self-explanation for those students who do not.

Since natural language understanding technology is still not powerful enough to reliably monitor self-explaining expressed verbally or as free text, the SE-Coach interface includes two alternative mechanisms: a masking interface to track the student's attention and a set of menu-based tools. The menu-based tools allow students to express self-explanations explicitly, if they want to. The masking interface allows the coach to measure how long the student focuses on a particular solution step. Such latency data helps the coach determine probabilistically when the student constructed a self-explanation mentally without entering it using the menu-based tools.

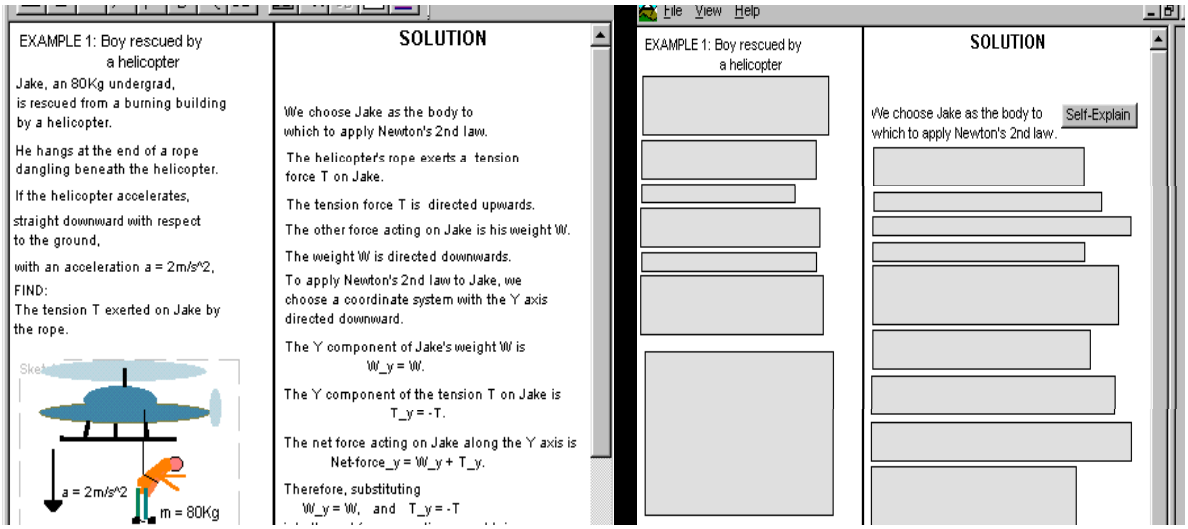

Figure 2: a physics example (on the left) as presented in the masking interface (on the right)

Figure 2 shows one of the SE-Coach's examples (on the left) as it is presented with the masking interface (on the right). Viewing a solution step in the example requires moving the mouse cursor over the box that covers it. When the student uncovers a step, a "self-explain" button appears next to it, as a reminder to self-explain. Clicking on this button generates more specific prompts that suggest two kinds of self-explanations known to be highly effective for learning (Renkl, 1997):

1. explaining an example solution step in terms of domain principles (*step correctness*);
2. explaining the role of a solution step in the underlying solution plan (*step utility*).

The SE-Coach's menu-based tools are designed to help students generate these two kinds of explanations, if they have trouble doing so by themselves.

To explain *step correctness*, the student can activate a Rule Browser (see Figure 3a) containing the hierarchy of physics rules represented in the Andes' knowledge base, and select the rule that justifies the uncovered solution step. To explain more about the actual content of a rule, the student can activate a Rule Template. The Template contains a partial definition of the rule that has blanks for the student to fill in (see Figure 3b). This definition is in terms of preconditions and consequences of the rule application, reflecting the definition of the rule in the Andes'

knowledge base. The student can complete the blanks in the definition through selection in associated menus of possible fillers (see Figure 3b).

To explain *step utility*, the student activates a Plan Browser, that displays a hierarchical tree representing the goal structure of the solution plan for a particular example. The student must try to select the plan goal that most closely motivates the uncovered solution step.

The SE-Coach's interventions during example studying include immediate feedback and explicit support on the self-explanation process. Immediate feedback is provided as a red or green marking on every student's entry in the self-explanation tools.

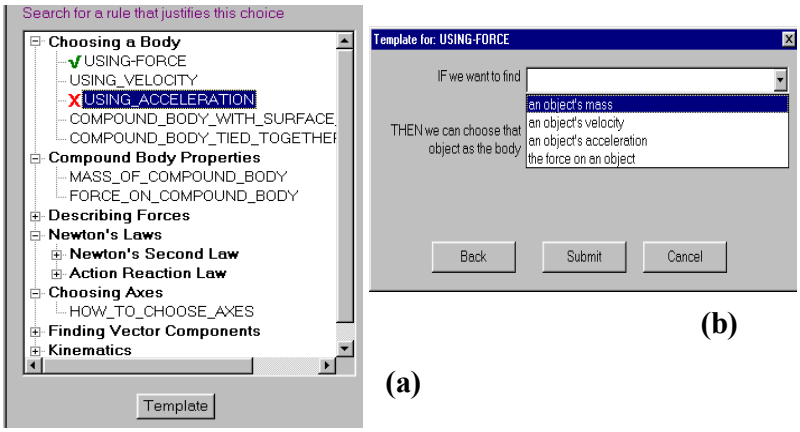

Figure 3: Rule Browser and a Rule Template

Explicit support for self-explanation is provided only when the student tries to leave the example. The support is given if the SE-Coach believes that the student would benefit from further explanation of some of the example's lines, either because the student does not have sufficient knowledge of the rules necessary to explain the lines or has not spent sufficient time reasoning about them. In this case, the SE-Coach generates a warning and highlights the appropriate masking interface boxes by coloring them pink. It also labels each box to suggest whether the student should explain step correctness or step utility or both, as an additional support for the student to explain those steps. The student is free to ignore the SE-Coach's suggestions and leave the example with no further action.

1.4 ANDES' BASIC APPROACH TO STUDENT MODELING

As mentioned earlier, because Andes allows students to pursue different correct solutions during problem solving, one of the main issues that Andes' student model needs to address is the assignment of credit for both plan recognition and assessment. That is, if a student action belongs to different solutions, which solution and corresponding student knowledge should be credited for it? Andes' student model handles this issue through a technique based on Bayesian networks. The technique was pioneered by the OLAE system for off-line assessment (Martin and VanLehn, 1995), and was extended to on-line assessment and plan recognition by the POLA student modeler, but only in prototype form (Conati and VanLehn, 1996a; Conati and VanLehn, 1996b).

In order to illustrate the basic technique, which is fundamental to the operation of the Andes' student model, consider a simple case where there are just two explanations for a particular student action. Suppose each explanation involves just one rule, and the preconditions of both

rules are satisfied directly by the state immediately prior to the student's action. This would be represented with the Bayesian network shown in Figure 4. The two parent nodes represent the rules, and the third node represents the student's action. The two parent nodes represent binary variables: true means the student has mastered the rule, and false means the student has not. Each rule node has a prior probability of mastery, which is the system's best estimate of the student's competence prior to seeing this action. The action node is also binary, and its value represents whether the action either has or has not occurred. The action node has a conditional probability table that says, in essence, if either parent rule is mastered, then the action will probably be executed; if both parent rules are non-mastered, then the action will probably not be executed. When the action is observed to occur, the action node is clamped to its "occurred" value, and the network is updated. This causes the two parent nodes to acquire a marginal posterior probability, which represents the system's new best guess about the student's competence. If one node has a much higher prior probability than the other then, by the Bayesian update rule, it will get most of the credit for explaining the student's action in that its posterior probability will increase more than the other node's.

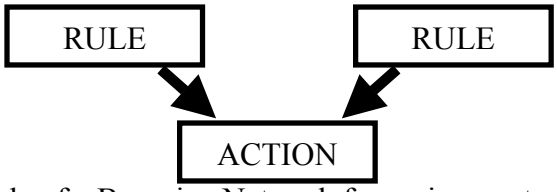

Figure 4: A simple example of a Bayesian Network for assignment of credit

This method of assigning credit is mathematically sound. Of course, whether it accurately predicts student competence depends on both the cognitive model *qua* network topology (e.g., are there really just two rules?) and on the prior and conditional probabilities in the nodes.

A Bayesian network can also handle the dual problem, assignment of blame. For instance, suppose that two rules must both be applied in order to produce a certain action. If the student cannot do the action, then the student is probably unable to use one or both rules. This would also be modeled by the simple network of Figure 4, except that the conditional probability table for the action node would be different: if *both* parents are mastered, then the action will probably be executed; if *either* is non-mastered, then the action will probably not be executed.[1] If the student is unable to perform the action, then the action node's value is clamped to "non-occurred" and the network is updated. This causes the posterior probabilities of mastery on both rules to be less than their prior probabilities. Because the student must be ignorant of at least one of the rules, the network has "blamed" both of them for the non-occurrence of the action. If one rule has a much lower prior probability of mastery, then it is more likely to be the missing rule, so it gets more of the "blame." This is a mathematically sound technique for solving the assignment of blame problem.

Although these illustrations have emphasized determining probabilities of mastery, which is the assessment part of the student modeling task, the same basic approach can be used for plan recognition. One represents possible student goals as parent nodes of observable actions, and the network will assign posterior probabilities to them that indicate the likelihood that those goals (intentions) underlie student actions.

---

[1] This is a simplification of Andes' actual network structure (described in section 2.2.2), but it reflects the same basic update mechanism.

Most importantly, such Bayesian networks cause assessment and plan recognition to interact seamlessly. Goals from explanations that involve probably mastered rules receive more credit for observed actions (and less blame for non-observed actions) than goals from explanations that involve probably unmastered rules. The mathematical soundness and the elegance of this approach motivated our adoption of it for Andes.

However, Andes is a real-world tutoring application, and thus we had to solve many technical problems to adopt the described approach for its student model. One was simply scaling the technology up. Many actions may be involved in solving a physics problem, and each action can require many rule applications to explain it. This makes the Bayesian network representing the problem solution quite large. Moreover, since the same rule can be used many times in solving a problem, each network can be highly interconnected. The size and topology of the networks could make them difficult to update in real time.

A second scaling problem is simply managing all the networks representing the many different exercises covered by Andes, and their relationships. Since the Andes system has over a 100 physics problems in it, it is not feasible to create each problem's network by hand. Moreover, the networks are all related because they all share the same rule nodes, yet it is clearly infeasible to use one monolithic network that spans all actions in all physics problems. Moreover, if even a small change in the knowledge representation is required, it can affect the networks of many problems.

In addition to these problems of scale, realistic student modeling requires interpreting much more than just problem solving actions made by students. In particular, Andes' Bayesian networks must deal with the following issues:

1. *Context specificity*: knowledge is sometimes acquired first in a more specific form then generalized, thus making near transfer (i.e., transfer to very similar application contexts) easier to obtain than far transfer (i.e., transfer to dissimilar application contexts) (Singley and Anderson, 1989). How can we track the generality of competence?
2. *Guessing*: some actions are easier to guess correctly than other actions. How should assignment of credit reflect this?
3. *Mutually exclusive strategies*: some problems have multiple, mutually exclusive solution strategies. Should evidence that the student is following one strategy be interpreted as evidence that they are *not* following the other strategy?
4. *Old evidence*: how should evidence from earlier problems affect the interpretation of evidence from the current problem? If students learn a rule, then old evidence that they were ignorant of the rule should be ignored. Should forgetting also be explicitly modeled?
5. *Errors:* should errors of omission (missing actions) be interpreted the same way as errors of commission (incorrect actions)? When can we assume that a student does not know how to do an action?
6. *Hints*: when the student has received hints before entering a correct action, how much credit should the goals and rules that explain that action receive?
7. *Reading latency*: when students study examples, they pause longer as they read some lines than others, and this may be evidence of self-explanation. How can we properly interpret the latency of reading times?

8. *Self-explaining ahead*: some students prefer to self-explain solution steps before reading them in the example, while others prefer to read steps then self-explain them (Renkl, 1997). How should such preferences affect the interpretation of reading latencies?
9. *Self-explanation menu selections*: when students use menus to express self-explanations, they may make a few errors and get feedback from the SE-Coach on each, then enter a complete self-explanation. How should a sequence of such menu selections be interpreted for knowledge assessment?

Although Bayesian networks were adopted in order to handle the assignment of credit and blame problems, they simplified handling the problems listed above as well. We were able to express using Bayesian networks sensible policies for handling those issues. We do not know if the individual policies are correct – that would require testing each one in isolation – but the policies did function well enough as a group that students using Andes learned more than students using more conventional instruction, as we will illustrate in a later section.

The remaining sections first describe Andes' Bayesian networks in general, then as they are used for coaching problem solving and for coaching example studying. These technical sections are followed by a summary of our evaluation results, which have been published in more detail elsewhere. The final sections discuss what we have learned, especially about the interpretation issues listed above, and how Andes' solutions compare to others in the literature. Our overall claim is that Andes is an existence proof that sensible policies for handling tricky interpretation issues can be easily expressed in Bayesian networks and yet the networks can still be updated in real time even when brought to the scale required by an ITS that has been used daily by hundreds of students for most of a semester.

2 The Networks of Andes

One of the scaling problems that Andes' Bayesian approach to student modeling presented is how to efficiently build the networks to cover the many exercises that Andes provides. Rather than create a Bayesian network by hand for each problem and example, Andes' networks are created by running a problem solver that for each problem and example generates a data structure called a *solution graph*, which is then converted to a Bayesian network. Thus, Andes uses the *knowledge-based model construction* approach to building Bayesian networks that has been actively investigated in recent years (Martin and VanLehn, 1993; Breese, Goldman and Wellman, 1994; Huber, Durfee and Wellman, 1994; Mahoney and Laskey, 1998).

For problem solving, Andes' solution graph represents *all* the correct solution paths of a problem. For example studying, the solution graph represents the single solution path that an example describes. In the next sections we describe how the solution graph is created and how it is converted into a Bayesian network.

2.1 The Solution Graph Representation

The solution graph structure used by Andes' student model is generated prior to run time by a rule-based physics problem solver. The rules are being developed in collaboration with four professors from the U.S. Naval Academy (Schulze, Correll, Shelby, Wintersgill and Gertner, 1998). Andes' knowledge base is built in CLIPS and contains 540 rules that can solve about 120 mechanics problems. The problems cover the first 11 weeks of physics teaching at the Academy and include most topics in introductory mechanics.

Like other rule based ITSs (e.g., Anderson et al., 1995), Andes rules encode a solution approach that uses goals to focus the problem solving and qualitative reasoning to prepare for the more algebraic, quantitative reasoning. Figure 5a and Figure 5b show two sample rules involving goals, while Figure 5c and Figure 5d show two sample rules encoding qualitative physics principles.

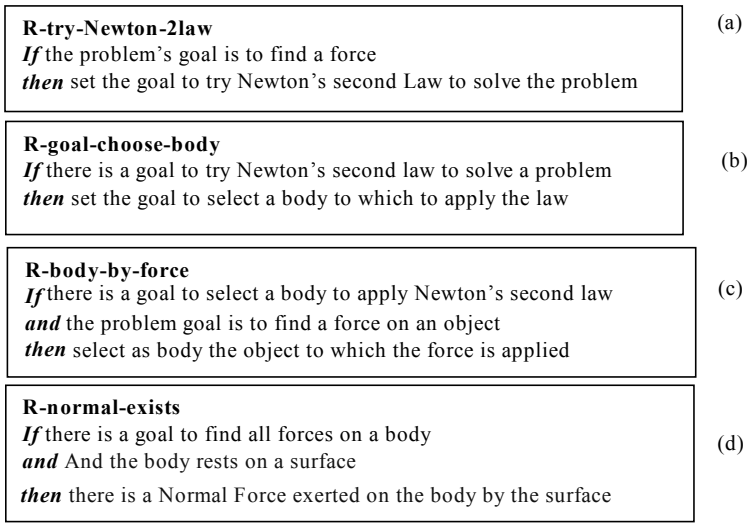

Figure 5: *Sample* rules in the Andes' knowledge base

When a new problem or example is created for Andes, the problem author must encode the *givens* for the problem in the language of the Andes problem solver. For instance, if the problem states that

```
A block (A) of mass 50kg rests on top of a table. Another
block (B) of mass 10kg rests on top of block A. What is the
normal force exerted by the table on block A?
```

the givens to the problem solver would include the proposition:

```
(SCALAR (KIND MASS) (BODY BLOCK-A) (MAGNITUDE 50) (UNITS KG))
```

meaning that there exists a scalar quantity of type mass which is a property of block A and has a magnitude of 50kg. The problem author also encodes the *problem goal*, specifying what quantity (or quantities) the problem asks the student to find the value of:

```
(GOAL-PROBLEM (IS FIND-NORMAL-FORCE) (APPLIED-TO BLOCK-A)
(APPLIED-BY TABLE) (TIME 1 2))
```

Starting from a problem's givens and goal, the problem solver iteratively applies rules from its rule set, generating sub-goals and intermediate results until all the sub-goals have been achieved. These goals and results are then written to a solution graph, along with the connections between them (e.g., if results 1 and 2 were both used by rule 3 to generate result 4, then 1 and 2, along with rule 3, are recorded in the solution graph file as being parents of 4).

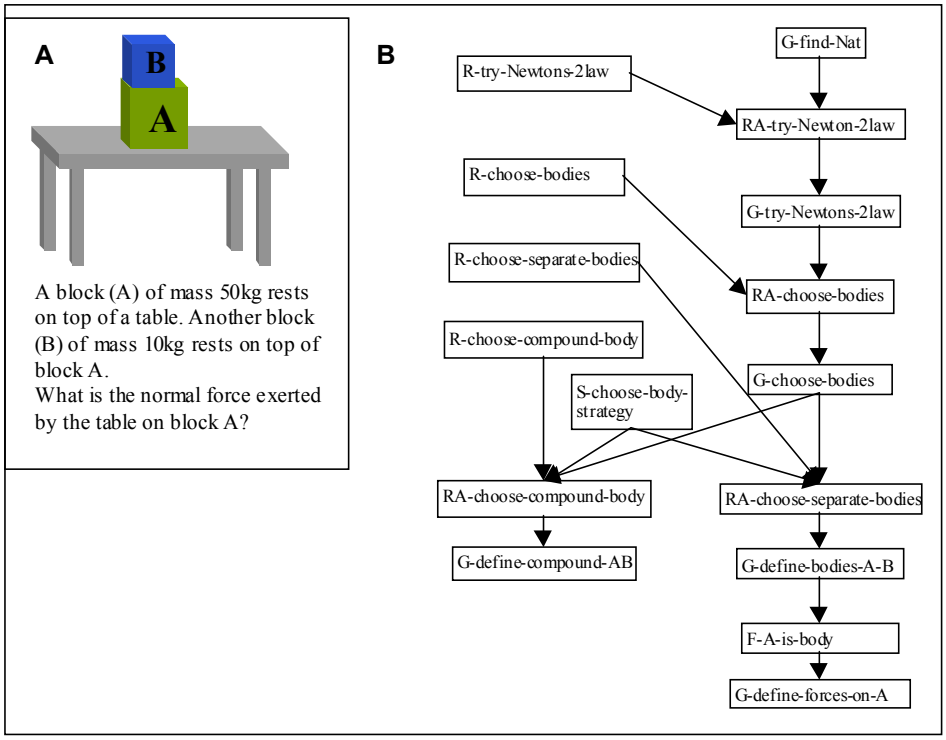

Figure 6: A physics problem and a segment of the corresponding solution graph. Some nodes are omitted for readability. Node prefixes (R, RA, G, F) indicate the type of the node, which will be discussed in detail in Section 2.2.2.

For instance, consider the example problem statement given above and shown in Figure 6A. The problem solver starts with the top-level goal of finding the value of the normal force N_{at}. From this, it forms the sub-goal of using Newton's second law to find this value. Next, it generates three sub-goals corresponding to the three high-level steps specified in the procedure to apply Newton's second law ($\Sigma \mathbf{F}_i = m*\mathbf{a}$):

1. choose a body/bodies to which to apply the law,
2. identify all the forces on the body,
3. write the component equations for $\Sigma \mathbf{F}_i = m*\mathbf{a}$.

From here, it continues to apply rules until it generates a partially ordered network of goals and sub-goals leading from the top-level goal to a set of equations that are sufficient to solve for the sought quantity. When the solution graph is generated for a problem, it encodes all the alternative plans and solutions that are acceptable to solve the problem, so that Andes can monitor and support the student in the development of any of these solutions. When the solution graph is generated for an example, it encodes only the solution that the example presents, because this is the solution that the student needs to understand.

Figure 6B shows a (simplified) section of the solution graph for the problem in Figure 6A, involving the application of Newton's second law to find the value of the normal force. Each node in the solution graph represents a particular type of information (fact, goal, rule, rule application), as indicated by the node prefixes. These node types will be discussed in detail in the next section. We will use the example in Figure 6 to show how the solution graph is

automatically converted into a Bayesian network, and describe the different types of nodes in the network and the relationships between them.

2.2 Structure of Andes' Bayesian Networks

Andes' Bayesian networks encode two kinds of knowledge: (1) *domain-general knowledge,* encompassing general concepts and procedures that define proficiency in Newtonian physics, and (2) *task-specific knowledge*, encompassing knowledge related to a student's performance on a specific problem or example.

The part of the Bayesian network encoding domain-general knowledge is built when the Andes knowledge base is defined, and its structure is maintained across problems and examples. At any time, the marginal probabilities in this network represent Andes' assessment after the last performed exercise.

The part of the network encoding task-specific knowledge is automatically generated from the solution graph of each problem or example when the student begins to work on it. When the student has finished the current exercise, the task-specific part of the network is discarded, and the updated domain-general marginal probabilities are "rolled-up" as priors for the next exercise. That is, Andes uses a *dynamic belief network* (Dean and Kanazawa, 1989; Russell and Norvig, 1995), to solve a second scaling problem of its student modeling approach: how to maintain a network of manageable size that still accurately represents the complete history of a student's interaction with the system. The following sections describe the structure of the networks, how roll-up is done, and how the conditional probabilities in the networks were determined.

2.2.1 The domain-general part of the Bayesian student model

The domain-general part of the network represents student long-term knowledge, via two different types of nodes: Rule and Context-Rule. Every node has two values, mastered or unmastered. For Rule nodes, mastery means that the student should be able to apply that piece of knowledge correctly whenever it is required to solve a problem, i.e., in all possible contexts. However, this simple mastered/unmastered distinction is only a crude model of human learning. Even when learning just a single fact, humans find it easier to remember the fact if the context of recall matches the context in which it was learned (Tulving and Thomson, 1973). When people learn a whole skill, many studies (reviewed in (Singley and Anderson, 1989)) have found that they can solve problems similar to the ones they were trained on (near transfer) long before they can solve problems that are dissimilar (far transfer). In short, as a general rule of thumb, knowledge is first acquired in a specific form then gradually generalized, an issue we referred to as *context specificity* in section 1.4.

Andes uses the distinction between Rule nodes and Context-Rule nodes in order to model the context specificity of student knowledge. Rule nodes represent a piece of knowledge in its fully general form. A Context-Rule node represents mastery of a rule in a specific problem solving context.

In order to represent the relationship between the general and specific versions of a piece of knowledge, every Context-Rule node has one parent, the Rule node representing the corresponding general rule (see Figure 6). This structure is similar to the one used by Hydrive (Mislevy and Gitomer, 1996), where the parent rules represented general knowledge (e.g.,

hydraulics) and the child rules represented applications of the knowledge in specific, named contexts (e.g., the landing gear; the cockpit canopy).

The conditional probability table of a Context-Rule given its parent Rule is defined as follows:

- *P(Context-Rule$_i$ = Mastered | Rule = Mastered)* is always equal to 1, because by definition *P(Rule = Mastered)* means that the student can apply the rule in every context.
- *P(Context-Rule$_i$= Mastered | Rule = Not mastered)* represents the probability that the student can apply the general rule in the corresponding context even if she cannot apply it in all contexts.

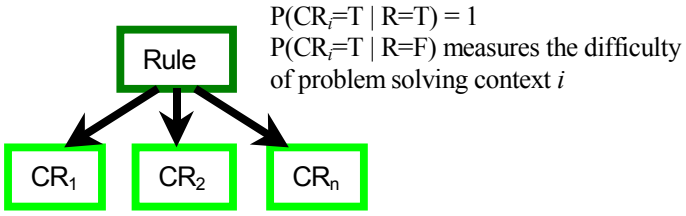

Figure 7: Relation between Rule and Context-Rule nodes

This network structure represents a hypothesis about transfer. Rule nodes in the domain-general part of the network are never directly observable, so sibling Context-Rules are always conditionally dependent. This means that user interface actions that increase the probability of mastery of one Context-Rule will tend to increase the probability of the parent Rule and thus increase the probability of mastery of a sibling Context-Rule. This is an empirically testable hypothesis about physics learning. In particular, results on transfer could be used to adjust the conditional probabilities in the Context-Rules, or even the topology of the network. Thus, the network provides a particularly simple, precise representation of a "mini-theory" of transfer, which invites testing but does not replace it.

2.2.2 The task-specific part of the Bayesian student model

The task-specific part of the Bayesian student model contains Context-Rule nodes and four additional types of nodes: Fact, Goal, Rule-application and Strategy nodes. All the nodes are read into the network from the solution graph for the current problem or example when the student opens it. The structure of the solution graph already encodes the inferential structure of the problem solutions and therefore is preserved isomorphically in the Bayesian network. For instance, a segment of the Bayesian network created for the problem in Figure 6A corresponds exactly to the solution graph segment in Figure 6B, where Context-Rule, Fact, Goal, Rule-Application and Strategy nodes are labeled respectively by the prefixes *R-*, *F-*, *G-*, *RA* and *S-*.

2.2.2.1 FACT AND GOAL NODES

Fact and Goal nodes (collectively called Proposition nodes) look the same from the point of view of the Bayesian network. They both represent information that is derived while solving a problem by applying rules from the knowledge base. The difference between Goal and Fact nodes is in their meaning to Andes' help system: the probabilities of Goal nodes will be used to construct tutorial interventions focused on the planning of the solution, while the probabilities of Fact nodes will be used to provide more specific interventions on the actual solution steps.

Goal and Fact nodes have a binary value indicating whether they are inferable. If a node has the value T, it represents that the student either has already inferred that fact/goal or can infer it given her current knowledge.

Goal and Fact nodes have as many parents as there are ways to derive them. Thus, if there are two different rule applications that conclude the same fact, then the corresponding Fact node will have two parents. This is the basic structure that Andes networks use to handle the apportion of credit problem.

The conditional probabilities between Proposition nodes and their parents are described by a Leaky-OR relationship (Henrion, 1989), as shown in the lower part of Figure 8. In a Leaky-OR relationship, a node is true if at least one of its parents is true, although there is a non-zero probability β of a "leak," that the node is true even when all the parents are false.

The leak probability β is Andes' solution to the *guessing* problem mentioned in section 1.4. It represents the likelihood that the student obtains the proposition via guessing, asking a friend for help, or some other method that is not modeled by the network. The leak probability can be adjusted to represent how easy it is to correctly guess a fact. For instance, a fact that asserts that an object's acceleration is constant is easier to guess than one that asserts the acceleration's direction, because acceleration is either constant or non-constant, whereas there are many more possibilities for its direction. Thus, our Bayesian framework allows us to concisely express a precise hypothesis about guessing, although empirical work is required to test it.

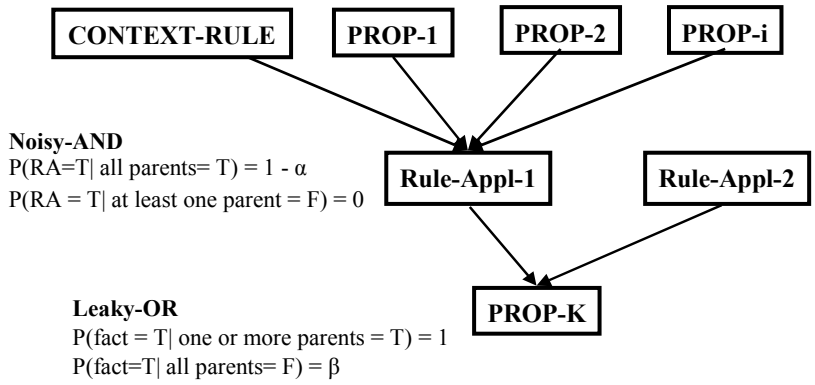

Figure 8: probabilistic relations among nodes in the task-specific part of the Bayesian network

2.2.2.2 RULE-APPLICATION NODES

Rule-application nodes connect Context-Rule nodes, Proposition nodes, and Strategy nodes (described later) to new derived Proposition nodes. The parents of each Rule-application node include exactly one Context-rule, some number of Proposition nodes corresponding to the Context-rule's preconditions (see Figure 8) and, optionally, one Strategy node.

Rule-application nodes have a binary value indicating whether the student is capable of applying the rule or not. A Rule-application node's value is T if the student has applied or can apply the corresponding Context-Rule to the propositions representing its preconditions. The probabilistic relationship between a Rule-application node and its parents is a Noisy-AND (Henrion, 1989). The Noisy-AND relation models the following assumptions

- in order to apply a rule to derive an element in a problem or example solution, the student needs to know the rule and all of the solution elements corresponding to the rule preconditions.
- there is a non-zero probability α, the *noise* in the Noisy-AND, that the student will not apply the rule even if the rule and all its preconditions are known.

The noise parameter α represents the fact that even if a Context Rule is mastered and all its preconditions are known, a student may fail to apply it sometimes, a phenomenon referred to as a *slip* (Norman, 1981). Thus, α should be viewed as part of the definition of mastery, because it represents the expected frequency of slips. Once again, the network allows us to precisely represent a testable hypothesis.

2.2.2.3 STRATEGY NODES

Because Andes allows students to follow different correct solutions to a problem, it needs to deal with the issue we labeled as *mutually exclusive strategies* in Section 1.4. Suppose we have the simple situation shown in Figure 9, in which there are two rule applications, both achieving the same goal as part of two different solutions. Suppose somewhere beneath the left rule application, an action occurs. This raises the posterior probability of both the left rule application and the goal. However, the credit flowing to the goal also raises the posterior probability of the right rule application. If the two rule applications are actually mutually exclusive means for achieving that goal, then raising the posterior probability of one should *lower* the posterior probability of the other. In order to model this, the network topology was augmented with Strategy nodes, extending a solution first proposed in (Conati and VanLehn, 1996a).

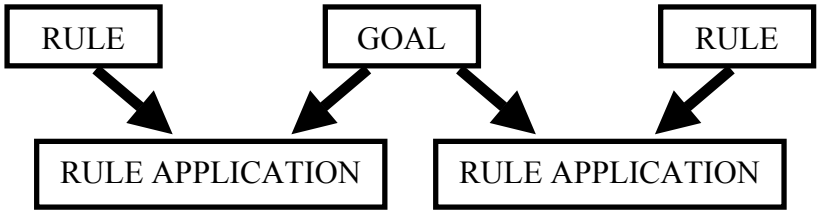

Figure 9: A simple Bayesian network illustrating the lack of mutual exclusion

A Strategy node is paired with a Goal node when there is more than one mutually exclusive way to address the goal. The children of Strategy nodes are the Rule-application nodes that represent the implementation of the different strategies. In Figure 6B, for instance, the Strategy node *S-choose-body-strategy* is paired with the goal of choosing a body, and points to the application nodes representing the decisions to choose block A and block B as separate bodies (node *RA-choose-separate-bodies* in Figure 6B) and to choose as a body the compound of the two blocks (node *RA-choose-compound-body* in Figure 6B).

A Strategy node has two or more values, one for each of its children. The prior probabilities of these values indicate the frequency with which students tend to choose the corresponding strategies. A rule application node that is the child of a Strategy node has a 1.0 probability of applying if the Strategy node has the value corresponding to it, and a zero probability of applying

otherwise. In short, the mutual exclusion of strategy node values is used to enforce the mutual exclusion of rule applications.

2.2.3 Calibration: Determining prior probabilities

Every node in a Bayesian network has a conditional probability table whose numbers must be determined before the network can be used. As the earlier discussion indicated, many of these numbers calibrate hypotheses about the context specificity of learning, guessing and so on, and thus the numbers ought to be determined via experimentation designed to test these specific hypotheses. Because the same parameters appear many times in the network, there are many fewer parameters to estimate than there would be if the network were not constrained by these hypotheses and every cell in every conditional probability table had to be determined separately. Moreover, the parameters have meaning apart from their occurrence in the network. For instance, α represents the frequency of slips for a mastered rule. When a parameter cannot be estimated from data, having a specific meaning makes it possible to base estimates for it on results from the literature. Even subjective estimates are probably improved when the parameters have well-defined meanings.

Although the hypotheses cover most of the conditional probabilities in the network, they do not cover the probabilities of the Rule nodes. Since a Rule node has no parents, its conditional probability table represents the prior probability of the student's mastery of the rule. Such priors are especially important, because, as illustrated earlier, when two rules compete to explain evidence, the rule with the higher prior probability receives more credit, all other things being equal. Similarly, rules with lower priors get more blame for missing actions. Data from multiple-choice tests were used to set the prior probabilities on 66 of the Rule nodes (VanLehn, Niu, Siler and Gertner, 1998). The other 474 rules did not apply in the problems on the test, so we could not set their prior probabilities empirically. Their priors were set to 0.5.

Even though we had to use subjective judgments to set most of the parameters in the network, this is still an advance over a purely heuristic approach since at least the structure of the network is theoretically grounded and the calculations are sound. We have conducted an evaluation of the sensitivity of the networks' assessment to its parameters, and the results are summarized in a later section.

2.2.4 Rollup: Summarizing past evidence

Although the interpretation of a student's actions should be affected by all the evidence seen so far from the student, it is not practical to use one huge Bayesian network that spans all the problems that the student has worked on. As mentioned earlier, we use *Dynamic Bayesian Networks* (Dean and Kanazawa, 1989; Russell and Norvig, 1995) to reduce the size of the network that models each individual interaction with Andes. The roll-up mechanism in dynamic Bayesian network allows one to periodically summarize the constraints imposed by older data, then prune away the network that interpreted that data. In particular, Andes keeps the structure of the domain-general part of the network intact over all exercises, but it prunes away the task-specific part of the network as soon as that task is exited by the student. However, the posterior probability of each Rule node in the task-specific network for the last solved exercise is "rolled-

up" in the domain-general network, to become the prior probability of that Rule node in the task-specific network for the next problem or example that uses it[2].

If one had a vast Bayesian network where actions in all problems were children of the same rule nodes, then evidence from past problems would have the same weight as evidence from recent ones. However, students may learn or forget rules. Thus, recent evidence should count more than old evidence when determining which rules are currently mastered. For instance, one could use an explicit "window," so that only evidence that is recent enough to fit in the window are used to update the probabilities of mastery. Corbett and Anderson (Corbett and Anderson, 1995) have developed a simple model of learning that makes transition to rule mastery at time T a function of both the evidence and the probability of mastery at time T-1. Basically, given the same evidence, the probability of a rule being mastered at time T is higher if it was mastered at time T-1 than if it was not mastered at time T-1. This represents hypotheses about the relative rates of learning and forgetting. We achieve the same effect in Andes by using the simple roll-up mechanism of changing posterior marginal Rule probabilities to priors. This causes rules with high posterior probability of mastery at time T-1 to attract slightly more credit for the evidence at time T, thus causing them to have slightly higher probability of mastery at time T than they would have it they had had low posterior probability of mastery at time T-1 and the same evidence were present. Thus, the old evidence still has some influence, although the rise in probability of mastery is determined mostly by the strength of the new evidence. Mislevy and Gitomer (Mislevy and Gitomer, 1996) propose essentially the same solution to modeling learning, but appear not to have implemented it.

3 Probabilistic student modeling for problem solving

The preceding section discussed some basic aspects of Andes networks that are the same for both problem-solving and example-studying. This section discusses how the network is used during problem solving. It also describes a few structural details that are unique to the problem-solving networks. The next section presents a similar discussion for the example-studying phase.

3.1 Example of Propagation of Evidence in the student model for problem solving

In order to illustrate the operation of Andes' student model during problem solving, suppose that a student is trying to solve the problem in Figure 6A and that her first action is to select block A as the body. In response to this action, the fact node *F-A-is-body* is clamped to T. Figure 10 shows the marginal probabilities in the network before and after the action (probabilities on RA nodes have been omitted for simplicity). Before the evidence is entered, these marginal probabilities reflect the prior probabilities on the rule nodes (0.5 each), the high prior probability of the given Goal node (G-find-Nat), and the priors on the Strategy node, which indicates that the two strategies are equally likely *a priori*.

---

[2] This simple roll-up mechanism loses any dependency among rules encoded in the task-specific part of the network. For instance, if two rules both explain an action (as in Figure 4), they are conditionally dependent in the task-specific network. When the task-specific network is pruned away, the conditional dependency is lost. This loss of information problem can be addressed by determining the main conditional dependencies among pairs of Rule nodes with a Chi-squared technique, then installing new nodes in the domain-general part of the network, whose conditional probability tables encode the strengths of the dependencies (Martin and VanLehn, 1994). However, experimentation with OLAE suggested that in our task domain, the conditional dependencies are usually not very strong. Thus, Andes does not currently use this more sophisticated roll-up technique.

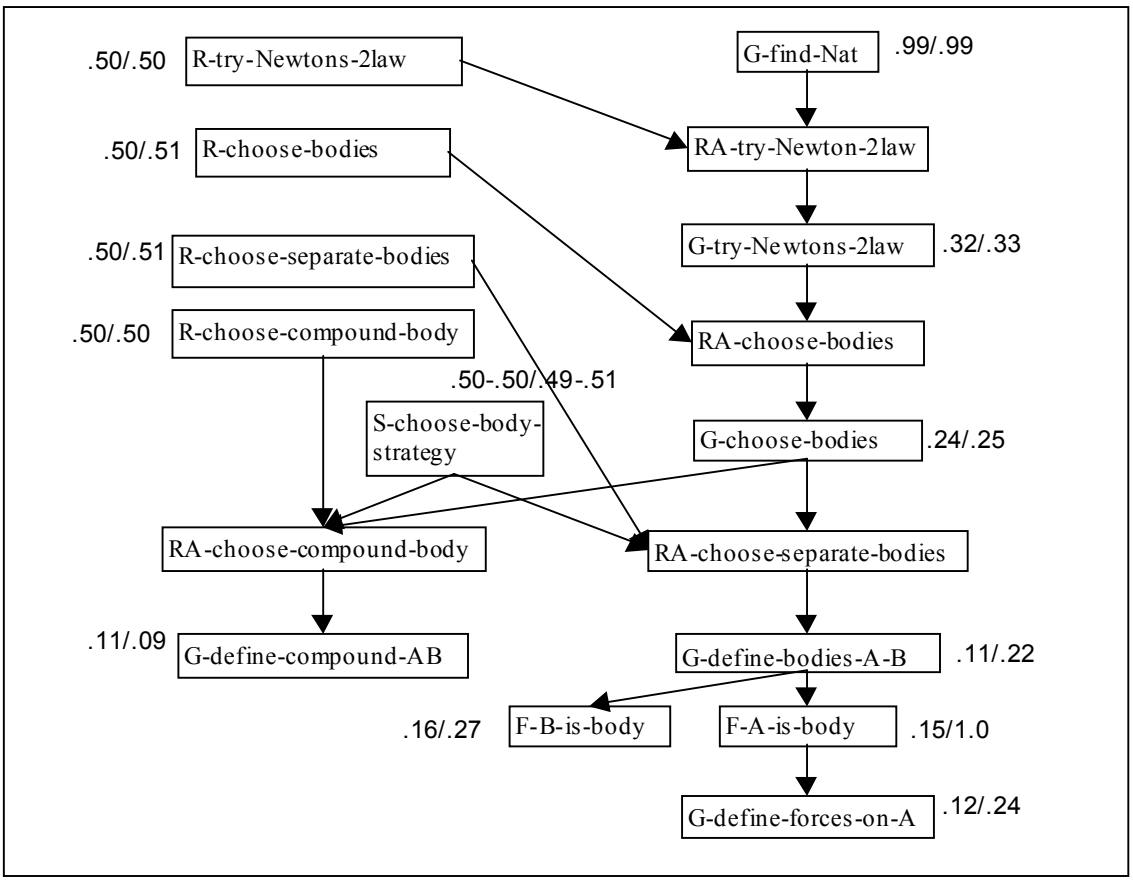

Figure 10: The network before and after observing A-is-a-body. The first number given for each node shows the probability of that node before observing A-is-a-body. The second number shows how the probability changes after observing A-is-a-body.

After the evidence is entered, the marginal probabilities of the non-given Fact and Goal nodes involved in deriving the observed action may seem rather low. However, for each proposition, the network evaluates the probability that it was derived vs. guessed. The more rules involved in deriving a proposition, the lower the likelihood that it was derived, since lack of mastery of any one of the rules would prevent its derivation. The likelihood that a proposition is guessed is proportional to the parameter β in its leaky-OR conditional probability table. In this example, all the rules have rather low prior probabilities, so their products are even lower, thus giving more credit to the possibility of guessing and keeping the posterior probabilities of the corresponding propositions low.

These remarks are just a crude summarization of what actually happens when the network is updated. Nonetheless, they suggest how complicated it would be to do the same interpretation without Bayesian computations.

In addition to changing the probability of the ancestor nodes of an observed action, updating the network also changes the probabilities of non-ancestor nodes, thus predicting what inferences the student is likely able to make after that action. In particular, the relationship enforced by the Strategy node *S-choose-body-strategy* on its children will slightly diminish the probability of the Goal node *G-define-compound-AB* and increase the probability of the Goal node *G-define-bodies-A-B*. This last change will propagate downward to increase the probability of the Fact

node *F-B-is-body,* while the increased probability of the Fact node *A-is-a-body* will slightly increase the probability of the Goal node *G-define-Forces-on-A*, as shown in Figure 10. Using the sound Bayesian propagation algorithm to predict the probability of student inferences is one of the several advances that Andes' student model brings to the basic Bayesian approach adopted by its predecessors OLAE and POLA. It provides the problem solving coach with a more principled, albeit still heuristic, mechanism to provide its help, as we will illustrate in a later section.

3.2 INTERPRETATION PROBLEMS THAT ARE UNIQUE TO THE PROBLEM SOLVING COACH

3.2.1 Errors of omission and errors of commission

Whenever a student enters a correct action, the corresponding Fact node is clamped to T, indicating that the student is capable of doing that action. What should the tutor do when the student's action is incorrect (an error of commission) or missing (an error of omission)? Should a Fact node be clamped to F?

Let us first consider the interpretation of errors of omission. If the tutor required that a certain action be done, and the student did not do it, then the Fact node corresponding to that action should be clamped to F. However, due to Andes' philosophy of allowing students to perform steps in any order and even to skip steps, there are only a few actions that Andes explicitly requires. In most cases, if the student has not done an action, she may still be able to do it. Indeed, she may even have done the action mentally or on a piece of scratch paper. Thus, Andes rarely clamps nodes to false due to missing actions.

For errors of commission, where the student makes an incorrect action, there are two cases. If the error implies that the student disbelieves a certain correct fact, then the corresponding Fact node is clamped to F. For instance, if the correct fact is that a vector is horizontal, and the student draws the vector pointing straight up, then the student undoubtedly believes that the vector is not horizontal so Andes clamps the horizontal Fact node to F. This first case, where the error implies disbelief in a specific correct fact, is relatively uncommon. The more common case is that the student's error is not inversely associated with a correct Fact. For instance, if a student enters an incorrect equation, it is often not clear what correct equation the student is trying to enter. A contradiction must be quite blatant (e.g., a vector cannot point both horizontally and vertically at the same time) before we can confidently infer that belief in one side of the contradiction implies disbelief in the other side. Thus, only in a few cases will Andes clamp nodes false due to errors of commission.

3.2.2 Updating the student model after a hint

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

This problem has been solved differently in different ITSs. For instance, the CMU Cognitive Tutors (Anderson et al., 1995) do not count actions that are preceded by hints for the purposes of knowledge tracing. The probabilities of mastery rise only when the student enters an action

without help from the tutor. Perhaps the most elaborate and potentially accurate method of interpreting hints is used by the SMART ITS (Shute, 1995), which uses a non-linear function derived from reports by experts to boost the level of mastery by different amounts, depending on the specificity of the hint and the level of mastery before the hint was given.

Our approach implements a simple theory of hints directly in the Bayesian network. The theory is based on two assumptions: first, hints are worded so as to remind the student of the requisite knowledge, rather than teach it. (Teaching missing pieces of knowledge is handled by the Conceptual Help system (Albacete and VanLehn, 2000), which is not discussed here.) Thus, procedural hints do not directly cause students to master knowledge. Second, a strong hint increases the chance that the student can guess the next action rather than derive it from her knowledge. Thus, a hint can cause an entry directly. In other words, hints affect actions directly but not domain knowledge.

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

3.3 USING THE NETWORK TO GENERATE HELP

Although the focus of this paper is on the construction of Andes' student model, it is worth a moment to explain how the student model is used to provide support during problem solving. Because Andes' policy is to help students solve problems their own way, rather than in a predefined optimal way, when a the student asks for help Andes needs to figure out what goal the student is probably trying to achieve (plan recognition) and which is the action the student is stuck on because of lack of the appropriate knowledge (action prediction).

Thus, after a help request, the procedure for selecting a hint topic (Gertner, Conati and VanLehn, 1998) starts from the most recently observed node in the Bayesian network and searches upward, looking for a goal that the student is most likely to be pursuing. It then searches downward from that goal node, looking for a rule application with a low probability of being performed. Such a rule application is probably where the student is stuck for lack of knowledge, so it becomes the focus of the coach's hint.

The search through the network is necessary because the semantics of a T value on a Goal node do not indicate whether the goal has been achieved yet or whether the student intends to achieve that goal next. It only indicates that the goal can be derived by the student at this time. As a consequence, when Andes is searching for a goal that the student is probably trying to achieve, it must explicitly check whether all the actions beneath the goal have probably been done. If so, that goal has probably been achieved already, and Andes should search for a different goal. This is an uncertain decision, as students are allowed to skip actions and do them mentally. Thus, even if some of the actions beneath a goal have not actually been done, the goal might still be

considered achieved by the student. This uncertainty is currently handled by heuristics in the search algorithm rather than the Bayesian network itself.

The Bayesian network is also used to decide whether the student should be given instruction on a rule. If Andes selects a certain rule application for a hint, and the posterior probability of its Rule node is below a set threshold, then it calls another coaching module, the Conceptual Helper, to give a mini-lesson on that rule (Albacete and VanLehn, 2000). Otherwise, it just gives hints designed to jog the students' memory and refocus their attention.

The posterior probabilities on the rules could also be used to guide the selection of exercises for the student. In particular, Andes could choose a problem that involves only a few rules that the student has not yet mastered, on the theory that such problems would be challenging but not too difficult, and thus would maximize the change of learning the unmastered rules. However, when Andes was evaluated at the US Naval Academy, we decided to have Andes assign the same problems to all students so that the evaluation would be more controlled.

4 Probabilistic student modeling for example studying

In order to help students study examples more effectively via self-explanation, the Andes' student model for example studying needs to assess how well a student is self-explaining the current example and to identify those parts that may benefit from further self-explanation.

This modeling task involves challenging new student modeling issues. First, the student model must be able to assess whether and how students self-explain even when the students do not build their explanations with the SE-Coach interface. This must be done using only limited data on students' attention and estimates of the students domain knowledge. Second, because some students tend to "self-explain ahead" parts of the examples that they have yet to uncover (Renkl, 1997), attention to a given example line can indicate not only self-explanation of that line, but also of subsequent lines. Third, because the SE-Coach menus provide strong scaffolding to build self-explanation, it is unclear how much credit a student's physics knowledge should get for a correct self-explanation that the student produces through these menus. These are the issues that were labeled, respectively, *reading latency*, *self-explaining ahead* and *self-explanation menu selection* in section 1.4. In the following sections, we describe how we extended Andes' Bayesian networks for problem solving to handle the additional interpretation challenges related to these issues.

4.1 The Bayesian network for example studying

The task-specific part of the Bayesian network for example studying is derived from an example's solution graph using the same mechanism that we described in section 2.2. For the SE-Coach, this network represents a model of correct self-explanation (SE model) for that example. In particular, because Rule-application nodes in the Bayesian network encode how solution steps in the example derive from physics and planning rules, they represent exactly the self-explanations for correctness and utility that the SE-Coach targets, and that we introduced in Section 1.3:

- how a solution step can be explained in terms of domain rules (step correctness)
- what goal each solution step achieves in the example solution plan (step utility).

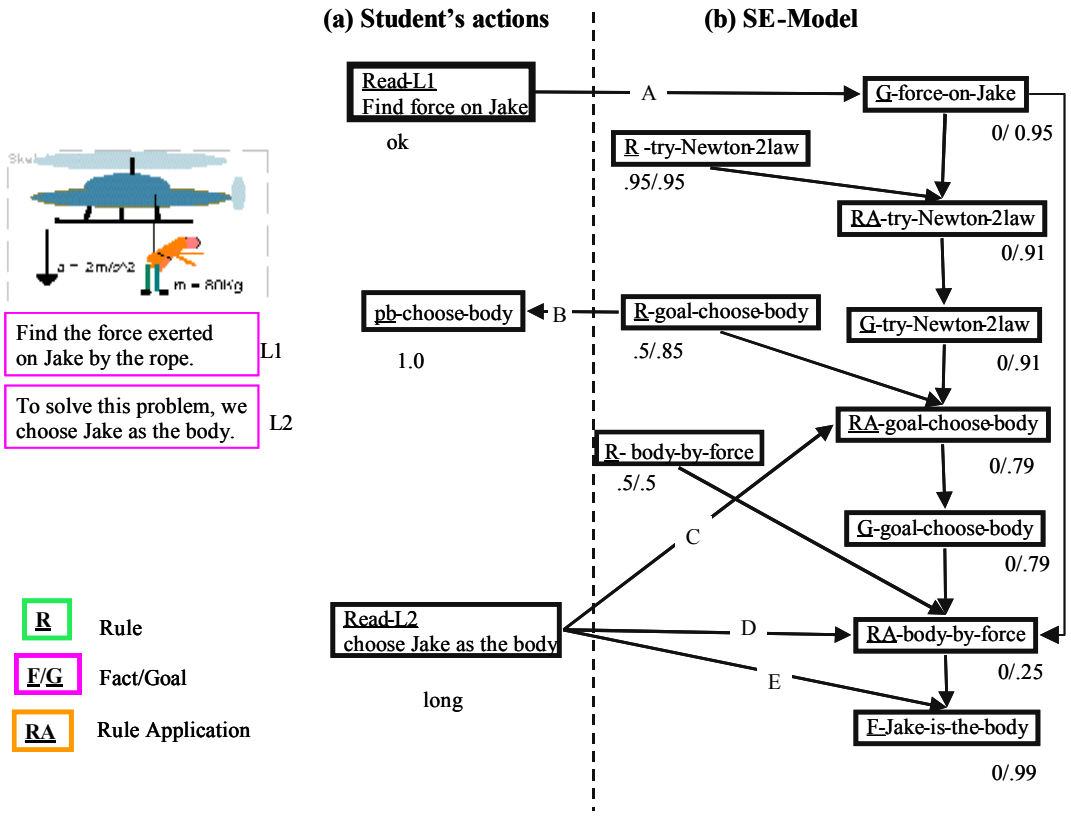

Figure 11: Segment of student model for the portion of example shown to the left. Part (b) of the figure shows the model structure before any student's action. The two numbers under each node indicate the node probability before and after the student's actions, represented by the nodes in part (a). The letters labeling some of the links are used to refer to the links in the text.

For instance, Figure 11b shows the segment of SE model for the part of the example shown on the left of the figure. The Rule-application node *RA-body-by-force* in Figure 11b encodes the explanation that Jake is chosen as the body because a physics rule says that if we want to find a force on an object, that object should be selected as the body to which to apply Newton's 2nd law. The node *RA-goal-choose-body* encodes the explanation that choosing Jake as the body fulfills the first subgoal of applying Newton's 2nd law, i.e., selecting a body to which to apply the law.

This interpretation of the task-specific part of the Bayesian network as a model of correct self-explanation for the current example requires a semantics that differs from the semantics in the networks for problem solving. The Boolean values of Proposition nodes in the Bayesian networks for example studying represent the probability that the student is aware of the corresponding facts or goals. The Boolean values of Rule-application nodes represent the probability that the student has self-explained the corresponding derivations. In particular, the probability

$$P(RA=T \mid \text{all parents} = T) = 1 - \alpha \qquad \text{(eq. 1)}$$

(where α is the noise parameter in the Noisy-And relation for Rule-application nodes, described in Section 2.2.2.2) is used in the student model to address the issue of *self-explaining ahead*.

This probability represents a student's tendency to self-explain an inference as soon as she has the knowledge to do so (Renkl, 1997). Hence, a student who tends to self-explain ahead will be characterized by a low α in the Noisy-AND relationship, while a student who tends to self-explain a line only upon reading it in the example will be modeled by a high value of α. Although the value of α generally depends on the student only,[3] it can sometimes also depend on the particular Context-rule in question. For instance, most students taking introductory physics understand whether they need to apply Newton's 2nd law as soon as they read a problem's statement. Thus, the Context-rule *R-try-Newton-2law* in b is generally associated with a low α.

Prior probabilities of root Proposition nodes in the Bayesian network, representing the example givens, are set to 0 until the student starts reading the example text (see for instance the first number under the node *G-force-on-Jake* in Figure 11b). As a student performs reading and self-explanation (SE) actions in the SE-Coach interface, the initial Bayesian network is dynamically updated with nodes representing these actions. Figure 11a, for instance, shows the nodes that are added to the network after a student

1. viewed the example line "*Find the force exerted on Jake by the rope*" long enough for reading it (node *Read-L1*);
2. viewed the line "*To solve this problem, we choose Jake as the body*" somewhat longer (node *Read-L2*).
3. self-explained "*To solve this problem, we choose Jake as the body*" with the Plan Browser (node *pb-choose-body*).

The second number under each node in Figure 11b shows the updated probability of that node after the student's actions in Figure 11a. In the next sections, we describe how Andes student model uses the Bayesian network formalism to interpret reading time and use of the self-explanation menus.

4.1.1 Interpreting the student's reading time

As mentioned earlier, one of the main problems for the SE-Coach student model is how to interpret student reading latency to assess self-explanation. The basic hypothesis we implemented in the student model is that the probability of self-explaining a rule application is a function of both the time a student spends studying an example line derived from that rule application and the probability that the student knows the rule and its preconditions. This section describes how that hypothesis is encoded in the Bayesian network.

Read nodes (nodes with prefix Read in Figure 11a) are dynamically added to the student model to represent viewing example lines in the SE-Coach masking interface. The values of Read nodes are LOW, OK or LONG. They reflect the duration of viewing time, and are assigned by comparing the total time a student spent viewing an item (TVT) with the minimum time necessary to read that item (MRT). We currently compute the MRT for each item by assuming 3.4 words per second, which is the reading speed of an average-speed reader[4] (Just and Carpenter, 1986). Depending upon the result of the comparison, the value of a Read node can be

[3] This parameter could be set, for instance, by observing how a student studies a test example.

[4] More accurate values for this parameter could be obtained by explicitly testing the reading speed of individual students before they start using the SE-Coach.

LOW (TVT << MRT, time insufficient for reading), OK (TVT≈MRT, time sufficient for reading only) or LONG (TVT >> MRT, time sufficient for self-explanation).

Each Read node connects to the Proposition node reflecting the semantic content of the viewed item (e.g., link A and E in Figure 11). These links indicate that viewing time influences the probability of being aware of the related content. When a Proposition node has input from *both* a Read node and a Rule-application node (e.g., *F-jake-is-the-body* in Figure 11b), then a student can acquire the corresponding proposition *either* by reading it in the example *or* by deriving it from rules and previous propositions.

READ		**LOW**	**OK**	**LONG**
Rule application	*T*	1.0	1.0	1.0
	F	< 0.4	> 0.9	>> 0.9

Table 1: Conditional probability table for a Proposition node viewed by the student

This relationship is represented in the conditional probability table for Proposition nodes (see Table 1). If the parent Rule-application node is T (i.e., the student explained the corresponding derivation), the proposition node is T (i.e., known by the student). Otherwise, the numbers in the table indicate that the probability of knowing the Proposition node depends on reading time: it is small if reading time is LOW, high when the time is OK, and even higher when the reading time is LONG.

Some Proposition nodes in the Bayesian network may not be connected to a Read node, even after the student has viewed all the elements in the example solution. This happens when the worked out solution omits some of the solution steps. In Figure 11, for instance, the nodes *G-try-Newton-2law* and *G-goal-choose-body* cannot have any Read node pointing to them, because these goals are not explicitly mentioned in the example solution. A student can become aware of unmentioned propositions only by deriving them.

knows	**knows**	**READ**		
CONTEXT- RULE	**all preconditions**	*LOW*	*OK*	*LONG*
T	*T*	1 - α	1 - α	max {(1-α), 0.9}
otherwise		0	0	0

Table 2: Conditional probability table for a Rule-application node after a student viewed the related Proposition nodes

The fact that the student viewed an example item does not necessarily mean that the student self-explained it. However, the longer the student viewed an example item, the higher the probability that the student self-explained it, provided that the student has sufficient knowledge to do so.[5]

[5] This relationship between probability of self-explanation and viewing time relies on the assumption that viewing time reflects time spent thinking about a given example line. This assumption is of course not always true, especially because we are currently measuring viewing time indirectly through the time a line stays uncovered. The assumption

This relationship is encoded in the student model by linking the Read node that represents viewing an example line with the Rule-application nodes that represent self-explanation for correctness (e.g, link C in Figure 11) and self-explanation for utility (e.g., link D in Figure 11) for that line. The conditional probability table for these Rule-application nodes is modified to take reading time into account, as shown in Table 2. No matter how long the student attends to the derivation, it cannot be self-explained correctly if the student does not have the necessary knowledge of the corresponding Context-rule and its preconditions. Cases where such knowledge is missing are grouped together in the row labeled "otherwise" in Table 2. On the other hand, if the student has the necessary knowledge, the probability that proper self-explanation occurs increases with viewing time. If viewing time is LOW (i.e., insufficient for reading) or OK (i.e., sufficient for reading only), self-explanation for this line could only have occurred if a student reasoned forward from previous lines. The probability that this happens is given by equation 1, representing the student's tendency to self-explain ahead. Thus,

$$P(RA=T \mid Rule=T, \text{All preconditions}=T, Read \in \{LOW, OK\}) = 1-\alpha$$

as shown in Table 2. If reading time is LONG, the probability that the student self-explained becomes high.

4.1.2 Modeling student use of the self-explanation menus

Another action interpretation problem that the SE-Coach student model must face is how much credit should be given to student self-explanations built through the SE-Coach menu-based tools, and how to count the feedback the SE-Coach provides as students work through the menus. This section describes how this problem is addressed in the Bayesian network.

Nodes representing the student's self-explanation actions (SE nodes) are dynamically added to the Bayesian network. For instance, the node *pb-choose-body* in Figure 11a represents the action of using the Plan Browser on the line "*To solve this problem we choose Jake as the body*." As we described in Section 1.3, the SE-Coach's templates reflect the content of the system's physics rules. Hence, template filling provides evidence of the student's understanding of the corresponding rule. This is encoded in the Bayesian network by linking the SE node for a template filling action with the Context-rule node corresponding to that template's content. Similarly, the selection of goals in the Plan Browser provides evidence of how well a student understands the mapping between solution steps and the underlying solution plan. This is encoded in the Bayesian network by linking Context-rule nodes establishing goals in the SE model with the SE nodes encoding goal selection in the Plan Browser (see link B in Figure 11).

Even if a student has little knowledge of a rule, she may still be able to generate a correct self-explanation involving that rule by using the SE-Coach tools, because of the guidance that these tools provide. Moreover, the more guidance the tools provide in the form of negative feedback on menu selections, the less evidence there is that the student knows the relevant rule. To model this interpretation, we direct the link from the Context-rule node to the SE node (see Figure 11a) and use the entry P(SE = T | Context-rule = F) in the SE node conditional probability table to represent the probability that a student can complete a correct SE action without knowing the

---

would be more accurate if we had a way to track the student's gaze to measure the student's reading patterns more accurately, or if the probability of self-explanation was also conditioned on a node representing a student's tendency to self-explain, determined *a priori*. We plan to explore these extensions in future versions of the SE-Coach.

corresponding rule. This conditional probability is adjusted dynamically depending on the students' use of the menus. In particular, the higher the number of wrong attempts a student makes before generating a correct SE action, the higher we set P(SE = T | Context-rule = F). This implements a sensible interpretation policy, namely that such student behavior makes it more likely that the student achieved the correct action through random selection in the SE-Coach tools, rather than through reasoning.

4.1.3 Using the Student Model to Support Self-Explanation.

As we described in section 1.3, the SE-Coach delays its interventions until the student tries to leave an example. At that point, the marginal probabilities of Rule-application nodes in the student model represent the probability that the student has self-explained the corresponding derivations. If a Rule-application node has a marginal probability below a given threshold, the SE-Coach will suggest that the student explain the associated example line. This section describes the process in more detail.

When the student indicates the desire to leave the example, the SE-Coach performs the following steps:

1. Check if the student model contains Proposition nodes that both correspond to example lines and derive from a Rule-application node with probability below the threshold for self-explanation.
2. For each of these Proposition nodes:
   - Add the corresponding example line to the list of lines that the student should explain further (i.e., the list of lines that will have their cover boxes highlighted in pink in the interface).
   - If the low probability of the parent Rule-application node is due to low probability of a Context-Rule node, add a prompt to self-explain the line using the Rule Browser/Template (when the Context-Rule node represents a physics rule) or a prompt to use the Plan Browser (when the Context-Rule node represents a planning rule).
   - If the low probability of the parent Rule-application node is due only to the low value of a Read node, add a prompt suggesting to read the line more carefully.

As an illustration, consider Figure 11, which shows the probabilities in the student model after a student performed the two reading actions and the Plan Browser action whose nodes appear to the left of the dotted line. Suppose that the student tries to leave the example now. The only Proposition node that derives from a Rule-application node with low probability and that corresponds to an example line is *F-Jake-is-the-body*. Therefore, the SE-Coach adds the corresponding example line "*To solve this problem, we chose Jake as the body*" to the list of lines to be further explained. Because the low probability of *RA-body-by-force* is due to low probability of the Context-rule *R-body-by-force*, the SE-Coach adds a prompt to use the Rule Browser/Template to self-explain the line. If the Context-Rule node *R-goal-choose body* also had low probability, the SE-Coach would add a prompt to self-explain this line using the Plan Browser. On the other hand, if both rule nodes had high probability while the value of node Read-L2 had been low, the SE-Coach would add only a prompt to read this line more carefully.

If the SE-Coach did not use the Bayesian network, it would have to make many more suggestions. It would probably suggest re-reading all lines whose latency was normal or below

normal. It would probably suggest using the self-explanation menus for all lines that have not yet been explicitly self-explained. With the Bayesian network, the suggestions of the coach are much more focused, and thus more likely to be useful to the student, and thus more likely to be acted on by the student.

5 Evaluations of Andes' student models

In this section, we first summarize the results of various evaluations that provide evidence of the effectiveness of Andes' student model for problem solving. Next, we describe an evaluation of the student model for example studying.

5.1 Evaluation of the student model for problem solving

Evaluation of a statistical approach to student modeling can use techniques from both disciplines of machine learning and user modeling (Zukerman and Albrecht, 2001). From a machine learning perspective, we are interested in evaluating the accuracy of the model's predictions on a known test set of data. User modeling evaluations focus on the accuracy of the model in predicting or inferring the state of real users. There is little consensus on a methodology for performing such evaluations, and very few are reported in the literature. We have performed a machine learning style evaluation of Andes' student model for problem solving (VanLehn and Niu, 2001) and, although we still don't have a formal user-modeling-style evaluation, we have empirical results showing that Andes enhances student learning, which provides indirect evidence of the model's effectiveness.

In the next two sections we summarize the results of the machine learning style evaluation and of the evaluations with real students.

5.1.1 Machine Learning Style Evaluation

In the machine learning style evaluation of Andes (VanLehn and Niu, 2001), a set of *simulated students* were created with predefined knowledge profiles, and these students' solutions to the Andes physics problems were simulated. The student model was run to determine if it could accurately detect the knowledge profile of the simulated students. Unlike some machine learning evaluations, where many researchers use the same data to evaluate their software, Andes can only be compared to different versions of itself. Thus, this evaluation concentrated on varying numerical parameters (e.g., the prior probabilities) and structural features (e.g., whether students are required to correct their errors).

The results show that Andes' assessment, with its normal parameter settings and structural features, correctly determined whether a rule was mastered or unmastered about 65% of the time, aggregating over all synthetic students and all evaluation conditions. There were two major impediments to increased accuracy.

The first impediment was that not all the inferences that students can make are visible on the user interface. For instance, when a student adopts a goal, there is no way to tell Andes about it. If every goal or fact node in the Bayesian network had a user interface action associated with it, then Andes' accuracy would rise from 65% to 75%.

The second impediment is that Andes does not require students to make entries even when its interface allows doing so. For instance, even if the student infers that the velocity of a body is

zero, the student doesn't have to write down the equation V=0. If Andes could assume that missing actions implied missing inferences, then its accuracy would rise from 65% to 70%.

If both impediments were removed, then Andes' accuracy would rise from 65% to 95%. In contrast, varying other parameters and features (e.g., prior probabilities, slip and guess parameters) had relatively little influence on accuracy.

These results suggest that Andes is doing a good job of student modeling given the amount of information that the user interface makes available about student reasoning. In particular, all the subjective probabilities that we had to include due to the lack of empirical data are probably not hurting Andes' accuracy much, as it proved not to be sensitive to these numbers.

The only way to do better appears to be requiring the student to display more of their thinking, but this would not only slow them down and increase the time to learn the user interface, but it could make Andes so tedious to use that students simply would not accept it.

5.1.2 Evaluation with real students

In the Fall of 1999 we performed a summative field evaluation of Andes at the Naval Academy (Schulze et al, 2000). Andes was used for four weeks by 173 students in eight sections of the Naval Academy's introductory physics course. At the end of this time, they were given a midterm exam covering material that was taught by Andes (and by the instructors during course lectures and sections). The students' performance on the midterm was compared to a control group of 162 students whose sections did not use Andes. The results of this comparison were encouraging. Students who used Andes performed about a 1/3 of a letter grade better on average than students who did not use Andes ($T(334)=2.21$, $p=0.036$, two-tailed).

Andes was also evaluated in the fall of 2000 (Shelby, Schulze, Treacy, Wintersgill and VanLehn, 2001). This version of Andes was essentially the same as the one evaluated the preceding year. However, it covered more chapters of mechanics, had more problems per chapter, included some more difficult problems, and required the students to draw vectors explicitly (in the previous version, students could define variables representing vector quantities without actually drawing the vectors). The experimental design was the same, with 140 students in the Andes condition and 135 students in the control condition. Students in the Andes condition scored significantly higher on the post-test ($T(274)=7.74$, $p<0.00001$, two-tailed). The effect size was 0.92 of a standard deviation, which at the US Naval Academy, corresponds to raising the students' grade by about one letter (e.g., from C to B). This effect size compares well with other ITS, many of which also have an effect size of about 1.0 (Shute & Psotka, 1996).

The Andes design was based on the hypothesis that instruction could be improved by simply coaching students as they did their homework and making no other changes in the course. This hypothesis appears to have been borne out. Although the US Naval Academy instructors collaborated with us in designing the system and were thus familiar with it when they began to use it in their classes, they used the same textbook, the same labs and almost the same lectures as were used in the Control condition and the rest of the course. In contrast, many other ITS that have also achieved a 1 standard deviation effect size required instructors to use a new curriculum and sometimes even a new textbook (e.g., Koedinger, Anderson & Mark, 1995; Corbett, McLaughlin & Scarpinatto, 2000).

Although learning gains are the traditional measure of success for educational software, an arguably more important measure is voluntary acceptance by both instructors and students,

because if either refuse to use the software, then it doesn't matter how much the software improves learning. As just discussed, Andes seems to work well even when instructors change very little in their course—in particular, they can continue to use their old textbook and lecture notes. This should increase instructor acceptance. Student acceptance was tested in the fall 2000 US Naval Academy evaluation. In the physics courses at the US Naval Academy, doing the assigned homework usually is not mandatory. Moreover, students in the Andes conditions were encouraged to use Andes, but they could use paper and pencil if they wanted. Thus, the number of problems done on Andes is a simple measure of student acceptance, because if students found it too difficult to use relative to its perceived benefits, then they would switch to pencil and paper or just not do their homework. From automated analyses of student log files, it was found that students solved 50 out of the 60 problems assigned on Andes. This is encouraging, but unfortunately we lack standards to compare this against, as we have no similar measure for the Control condition or from other educational software.

A final type of evaluation that may be done for a student model that does plan recognition is to look at how accurate the system is at inferring the students plans. Both positive learning gains and the students' acceptance of Andes provide indirect evidence that its student model for problem solving does quite a good job in predicting the students' intended goals, or at least that Andes' interventions based on the model's predictions are not disruptive or annoying for the students. Gaining more direct evidence of the student model predictive accuracy is hard, because one must have a standard for judging what the students' goals actually were. We attempted to overcome this difficulty by using an approach that involves comparing the predictions produced by a user model to subjective human judgments of the user's goals. Using log files from the 1999 evaluation, we randomly selected episodes where the students asked Andes for help. We generated a snapshot of the Andes screen just prior to its help message. We then gave these screen snapshots to expert physics tutors, and asked them to judge (a) the goal that the student was trying to accomplish and (b) what help they would give to this student. Our plan was to evaluate Andes using only snapshots where the experts agreed with each other. Unfortunately, the agreement among the experts was so low that there were too few snapshots left for evaluating Andes. We intend to repeat the study with a new version of Andes and a much larger sample of snapshots. We will also give the judges a summary of the tutorial session up to the point of the snapshot, since they said they sometimes needed that in order to make an informed judgment.

5.2 Evaluation of the student model for example studying

The SE-Coach student model guides the SE-Coach to elicit students' self-explanations on lines that are assessed to be problematic for them. Thus, an indirect way to evaluate the effectiveness of this student model is to verify how the SE-Coach interventions influence students' learning. We obtained preliminary results on this from a study that we conducted in our lab.

During the study, 29 subjects studied Newton's 2nd Law examples with the complete SE-Coach (experimental condition), while 27 subjects studied the same examples with the masking interface only and no coaching (control condition). All subjects took a pretest and a posttest consisting of Newton's 2nd Law problems. At the time of the experiment, all subjects were taking introductory physics at one of three different colleges and had started studying Newton's laws in class. In this section, we focus on results related to how the SE-Coach interventions, and thus the student model, influenced the performance of students in the experimental group. More

general data on the positive results we obtained on the difference between the performance of the experimental and the control groups can be found in (Conati and VanLehn, 2000).

Prompt Type	Max.	Generated	Followed
Use RuleBrowser/Templ.	43	22.6	38.6%
Use PlanBrowser	34	22.4	42.0%
Read more carefully	43	7	34.0%

Table 3: Percentage of SE-Coach prompts that students followed

Constraints on the study duration prevented us from initializing the student model with data on the students' initial knowledge and studying style. Hence, we assigned to most of Andes' rules a prior of 0.5, and we assumed that students were very unlikely to explain ahead ($\alpha = 0.98$) because other studies (Renkl, 1997) show that this is consistent with most students' behavior.

We computed from log data how often students followed the SE-Coach's adaptive prompts to further self-explain. The results are summarized in Table 3.

For each type of prompt, the table reports: (i) the maximum number of prompts that could appear in the interface for the three examples used in the study. These are the prompts the SE-Coach would generate if there was no student model. (ii) The average number of prompts adaptively generated by relying on the student model. (iii) The average percentage of these prompts the students followed. We then computed the correlation between the percentages of followed prompts and students' post-test scores, after regressing out pretest scores. All three measures significantly (or nearly significantly) correlate with post-test performance ($p = 0.056$, R^2=66% for Rule Browser/Template prompts; $p = 0.024$, R^2=64.5% for Plan Browser prompts; $p = 0.016$, R^2=63% for "Read more carefully" prompts). These data are consistent with the hypothesis that the adaptive prompts based on the student model effectively elicited self-explanations that improve students' learning, although further data should be gathered to control for other variables that might have caused the correlation, such as general academic ability or conscientiousness.

The correlation exists despite the fact that students, on average, followed less than half of the SE-Coach prompts (see Table 3). We conjecture two possible explanations for why students did not follow the interface prompts more extensively. The first is that the low accuracy of the initial parameters in the student model caused the model to underestimate the amount of spontaneous self-explanation that students generated. Thus, students rightly ignored those prompts asking them for redundant self-explanations. The second reason is that those students who tended to overestimate their understanding ignored the SE-Coach prompts, even when the prompts were justified. If this was the case, obligating the students to follow the SE-Coach suggestions would increase the effectiveness of the system. Further evaluations of the student model with more accurate initial parameters will clarify this issue.

We also examined the accuracy of the SE-Coaches assessment. We found an interaction between accuracy and when subjects had started studying Newton's Laws in their classes. In particular, we computed the correlation between posttest scores and the number of rules that reached high probability in the student model. The correlation is very low ($r = -0.03$) for the 17 subjects from classes that had started the example topic more than a week before the study

(early-start subjects) and it is higher (r = 0.33) for the 12 subjects from classes that had started just a few days before the study (late-start subjects). Since our data showed no significant differences in the two groups' initial knowledge or in how they used the interface tools, we hypothesize that the difference in the correlation exists because the SE-Coach examples were more challenging for late-start subjects and therefore they put more effort in reasoning and learning from the same SE actions. Hence, their long reading times and use of the menus really was associated with self-explanation, whereas similar behavior from the early-start students was not.

If this interpretation is correct, then it suggests that the students' learning stage should be taken into account when assessing knowledge based on SE actions. For instance, in the conditional probability table for SE nodes, the probability P(SE=T | Rule=F) (e.g. the probability of generating a correct SE action without knowing the corresponding rule), should be set to a higher value if a student has been working on the example topic for some time (like our early-start subjects), so that this student's knowledge receives less credit for her self-explanations.

6 Related Work

The development of techniques for reasoning under uncertainty that have appeared in the AI community since the mid eighties has facilitated a rapid growth of interest in probabilistic approaches to user modeling. As Jameson (1995) outlines, these user models tend to address three general purposes: (1) inferring the user's knowledge or general abilities (Mislevy, 1995; Petrushin, Sinitsa and Zherdienko, 1995; VanLehn and Martin, 1998), (2) recognizing the user's plans or goals (Charniak and Goldman, 1992; Huber et al., 1994; Conati and VanLehn, 1996b; Albrecht, Zukerman and Nicholson, 1999), or (3) predicting the user's inferences (Van Mulken, 1996; Conati and Carenini, 2001) and future behavior (Horvitz and Barry, 1995; Horvitz, Breese, Heckerman, Hovel and Rommelse, 1998; Zukerman, Albrecht and Nicholson, 1999). Depending on the purpose of the system, one of these functions may be treated as more important than the others. For instance, Charniak and Goldman are primarily interested in recognizing the plans of characters in a story, while Horvitz and Barry are interested mainly in how the user will respond to information presented by their system. In a student model that supports provision of help during a fairly unconstrained tutorial interaction all three functions are important and interrelated. Thus, the Andes' model combines all three of them into a single Bayesian network representing both the students' domain knowledge and also their past and future plans and goals.

Student modeling poses some unique challenges for probabilistic approaches due to the fact that students' knowledge changes over time as they learn. This makes it even harder to perform plan recognition and behavior prediction because the student model cannot assume, as many plan recognition systems do (Carberrry, 2001), that the user always has perfect knowledge of the available plans throughout the interaction. Both the models developed for the Carnegie Mellon Cognitive Tutors (Corbett and Bhatnagar, 1997) and for the SMART system (Shute, 1995) are designed to assess the student's knowledge as it evolves during the interaction with the tutor, and both of these systems have been favorably evaluated as accurately reflecting students' learning. While neither of these systems uses Bayesian networks, Reye (Reye, 1998) has shown that both models can be formulated as a dynamic Bayesian network with the same basic behavior. However, none of these models make use of probabilistic knowledge assessment to guide their inference of the student's goals, as Andes' student model does. While Shute's system does not need to infer students' goals, the cognitive tutors do. This is because, although they make the

students explicitly enter all their solution steps and thus reduce uncertainty about what the student is trying to do, their solutions don't have fixed step-ordering. Thus, like in Andes, when a student has asked for help, there are several possible correct actions that could be done next. The Cognitive Tutors resolve the ambiguity by using heuristics based on asking the student, on the location of the student's cursor (Anderson et al., 1995), or on the student's most recent action (Corbett, McLaughlin and Scarpinatto, 2000).

As we discussed in Section 3.3, Andes relies on the student model probabilities to predict which goal the student is probably focusing on. These predictions drive a heuristic algorithm for selecting the content of the tutor's hints. DT Tutor (Murray and VanLehn, 2000) extends the Andes network by adding decision nodes and utilities so that the choice of tutor action can be done completely probabilistically. In particular, the tutor uses the extended network to predict the student's reaction to proposed tutor actions, evaluates the utility of the resulting student, tutorial and dialogue states, then chooses the tutorial action that maximizes the expected utility. However, this improved reasoning comes at the cost of greatly increased network size and therefore reduced speed for updating the network.

While Bayesian networks have been by far the most popular formalism in recent work on numerical user models, other numerical methods have been used as well. In particular, Jameson (Jameson, 1996) describes a number of systems that use Dempster-Shafer theory (Carberry, 1990; Bauer, 1995) and Fuzzy Logic (Katz, Lesgold, Eggan and Gordin, 1992). A recent review article of the field (Zukerman and Albrecht, 2001) shows how work on statistical techniques in user modeling has broadened in scope in the last five years.

An important and difficult question with respect to probabilistic or numerical approaches to student modeling is how they might be evaluated (see Section 5 for a discussion of evaluations of the Andes system). Student models that focus on knowledge assessment may be evaluated by comparing their predictions of the student's knowledge to actual student performance on post-tests. The SMART student model (Shute, 1995), which uses a set of regression equations to update the estimate of students' mastery of each curriculum element, was evaluated and found both to accurately predict students' posttest performance and to contribute to impressive learning gains of greater than 2.0 standard deviations when used to guide the tutoring system's behavior. Similarly, the knowledge tracing component of the ACT Programming Tutor was found to accurately predict student post-test performance (Corbett et al., 2000), and to contribute to students' learning more and faster using mastery learning (Anderson et al., 1995). However, there are several problems with this approach. The student model assesses student competence on many individual pieces of knowledge, but the post-test yields a single score. This can allow systematic inaccuracies in the student model to remain undetected. To put it a little differently, when the outcome of the evaluation is a single number (the correlation between predicted post-test score and actual post-test score), how can one interpret its value? As Corbett et al. (2000) discovered, it takes substantially more work to discover why the correlations are low and make the appropriate changes to the student model to raise them. Moreover, as with any assessment technique, there are the classic issues of reliability and validity. For instance, a post-test can only sample the student's knowledge, which raises the issue of whether it is a "fair" sample of the competence in the task domain. VanLehn and Martin (1998) survey classic standards for evaluation of assessment systems, and indicate how OLAE (Andes' predecessor) fares. Lastly, the main problem with using post-tests for evaluation of knowledge assessment is that it confounds the assessment technology with the cognitive task analysis. As Corbett et al. (2000)

show, their cognitive model sometimes used one rule when subjects actually used two, for instance. Although this creates an inevitable inaccuracy in the assessment, it is a fault of the cognitive task analysis and not of the assessment technique per se. For all these reasons, evaluations with synthetic students (like the Andes' evaluation we discussed in section 6.1) are a very promising alternative to provide valuable insights on the details of a user-model performance.

User models that focus on inferring a user's goals or intentions tend to be more difficult to evaluate, since they require judging *a posteriori* what goals the user had at different times during the interaction. One approach that has been used so far is that of (Calistri-Yeh, 1991), who compared the predictions produced by his system to that of human judges. We used a similar technique to evaluate the prediction capabilities of Andes' student model for problem solving, as discussed in Section 5.1.2.

7 Discussion and Future Work

We have presented a probabilistic approach to student modeling that relies on Bayesian networks to deal with the high level of uncertainty inherent in providing tailored support for homework-related activities in Andes, an intelligent tutoring system for Newtonian physics. The homework-related activities that Andes supports include learning from worked-out examples in addition to problem solving, because example studying is one of the fundamental activities that complement classroom instruction.

Much of the uncertainty that Andes' student model faces is due to Andes' philosophy of giving students the initiative in the learning process. This means, for instance, that students have the freedom to experiment with different solutions and learn from their own mistakes, while Andes provides help and support when asked to but does not constrain students to follow a predefined solution path. Furthermore, Andes does not require students to always express all their reasoning explicitly.

This focus on student initiative involves a higher level of uncertainty than more constrained ITSs do (e.g., Anderson et al., 1995) for two main reasons. First, allowing students to follow multiple solutions forces Andes to confront the assignment of credit problem, which arises every time there is more than one explanation for a user's interface action. Second, not requiring students to make all their reasoning explicit can drastically reduce the amount of information the student model has for generating its predictions (the *bandwidth problem* discussed in (VanLehn, 1988)). For instance, when discussing the VanLehn and Niu (2001) evaluation of the Andes' student model, we noted that student modeling accuracy would increase from 65% to 95% if Andes required students to enter on the user interface all facts or goals as they were inferred. But Andes' example studying and problem solving components cannot feasibly require students to explicate all their thinking, because this would so greatly add to the students' burden that few students would be willing to use such a tutor, especially as they become more proficient in the instructional domain. Hence, Andes' student model will often have little information on the student's reasoning and thus must guess.

In this paper, we illustrated how we rely on the probabilistic reasoning framework of Bayesian networks to enable Andes to make as much of an educated guess as possible based on the available evidence. Given that one cannot have both freedom and convenience for students while also having highly accurate student modeling, it appears that our Bayesian student model has

allowed Andes to made an appropriate compromise between the two, in that Andes is indeed an effective tutoring system. Both the Problem Solving Coach and the SE-Coach cause learning gains, and the Problem Solving Coach appears to be acceptable to students in a field testing situation.

The key to Andes' success, as with all Bayesian network models, lies in accurately representing the probabilistic dependencies in the task domain. Andes uses a knowledge-based model construction approach to generate the part of the network that represents the probabilistic dependencies between domain knowledge and inferences in problem solutions. The principles and parameters used in creating chunks of Andes' networks are essentially little theories of how specific events relate to each other and to the user's knowledge and goals. We list here a brief summary of how these theories were encoded in the Bayesian network formalism, along with a few lessons we learned along the way.

7.1 A DEPLOYED, REALISTIC STUDENT MODEL BASED ON BAYESIAN NETWORKS

As we mentioned earlier, the main problem that Andes student model needs to address is the assignment of credit problem. The Bayesian solution that it adopts is built on the approach that OLAE pioneered for assignment of credit during off line assessment and that POLA extended to plan recognition. None of these Andes' predecessors went beyond the prototype stage. To scale up the basic approach for use by a fully fledged system deployed in the field, we had to add hypotheses about knowledge, learning and behavior that increased the realism of the student model. In all cases, a sensible hypothesis was posed and implemented in the Bayesian network, albeit without the benefit of empirical calibration to determine the relevant parameters. In particular, we have presented solutions to the following issues:

1. *Context specificity*: knowledge is sometimes acquired first in a more specific form then generalized, thus making near transfer easier to obtain than far transfer (Singley and Anderson, 1989). How can we track the generality of competence? Andes distinguishes Context Rules from Rules (see Section 2.2.1).
2. *Guessing*: some actions make it easier to guess correctly than other actions. How should assignment of credit reflect this? The parameter β in the leaky-OR gate (which defines the conditional probability between an action and the different ways to derive it) is higher when guessing is more likely to lead to success (see Section 2.2.2.1).
3. *Mutually exclusive strategies*: some problems have multiple, mutually exclusive solution strategies. Thus, evidence that the student is following one strategy should be interpreted as evidence that they are *not* following the other strategy. Andes student model uses Strategy nodes to enforce this behavior (see Section 2.2.2.3).
4. *Old evidence*: how should evidence from earlier problems affect the interpretation of evidence from the current problem? Andes rolls up the task specific network of a completed problem by copying the marginal probabilities on Rule nodes into prior probabilities (see Section 2.2.4).
5. *Errors:* errors of omission (missing actions) cause nodes to be clamped to False only when the action is required, which happens rarely since there are only few required actions in Andes. Errors of commission (incorrect actions) cause nodes to be clamped to False only when the incorrect action directly and blatantly contradicts the node's action (see Section 3.2.1).

6. *Hints*: when the student has received hints before entering a correct action, how much credit should the goals and rules that explain that action receive? An additional node is added to those representing alternative derivations of the action, so that the more specific the hint, the less credit those derivations receive (see Section 3.2.2).
7. *Reading latency*: when students study examples, they may pause longer as they read some lines than others, and this may be evidence of self-explanation. How can we properly interpret the latency of reading times? Andes uses Read nodes whose values record the observed latency and combines them with knowledge assessment to judge whether a student is self-explaining (see Section 4.1.1).
8. *Self-explaining ahead*: some students prefer to self-explain solution steps before reading them in the example, while others prefer to read steps then self-explain them (Renkl, 1997). Andes uses the noise parameter α to model how these two different studying behaviors affect the interpretation of reading latencies (see Section 4.1).
9. *Self-explanation menu selections*: when students use menus to express self-explanations, they may make a few errors and get feedback from the SE-Coach on each, then enter a complete self-explanation. How should a sequence of such menu selections be interpreted? Andes dynamically changes the conditional probability table of the SE nodes in order to reflect the amount of feedback the coach gives the student (see Section 4.1.2).

What we learned is that Bayesian networks, even when created by a rule-based problem solver, are like any other knowledge representation formalism in that one can fairly easily represent increasingly complicated and hopefully more realistic models. Like any other knowledge-based system, Andes should have its knowledge validated empirically, but this is unrealistic in most cases. We are fortunate to be able to evaluate the system as a whole with real students, as described in Sections 5.1.2 and 5.2. The best we can do for testing individual hypotheses is to use synthetic students, as described in section 5.1.1.

7.2 The need for plan recognition in unconstrained learning environments)

Although assessment is important in itself, because instructors often want to see just which rules a student has mastered, plan recognition is only useful inside the ITS as a decision aid. That is, it helps the ITS decide what tutorial action to make. In building a comprehensive student modeling framework that could support both assessment and plan recognition, we realized that they are sometime in conflict.

In Andes, plan recognition is necessary for the problem solving coach to select what step to suggest when a student asks for help. Because Andes wants to help students solve problems in their own way, rather than teach a problem solving plan of its own as other tutoring systems do, it must determine what goal the student is probably trying to achieve, and suggest the action the student cannot perform for lack of knowledge. This defines the kind of plan recognition that the problem solving coach needs.

As we discussed in Section 3.3, the semantics of the T values on proposition and rule application nodes in the student model for problem solving don't quite match the plan recognition requirements, because they only tell what the student can potentially derive, not what she has already derived or what she wants to derive next. In short, the problem is that Andes' nodes have the right semantics for assessment but the wrong semantics for recognizing the student's

intentions. Thus, the problem solving coach needs to resort to heuristic search to select a node for hinting, although the search is guided by the student model's prediction of what inferences the student can or cannot make.

One way to deal with this problem without resorting to heuristics as Andes does is to augment the network so that it can represent both competence and intentions (Murray and VanLehn, 2000). This requires Bayesian networks that have two nodes for each goal. One node has the same assessment-based semantics as Andes' goal nodes, and the other node indicates whether the goal is probably one that the student is currently intending to do. This allows the coaching system to perform hint selection by relying solely on the network probabilities, but unfortunately generates much larger student models that currently cannot provide the real-time responses that an Andes-like interaction needs.

Andes only needs to recognize student intentions because it wants to help students do what they intend to do. So one might wonder whether it is it really worthwhile to devise intelligent assistants that support students in fairly unconstrained pedagogical activities, considering the effort and challenges involved in building the student models suitable for these assistants. This question may be further justified by the fact that empirical studies with the CMU cognitive tutors, (which support more constrained interactions than Andes and therefore rely on less complex student models) have shown that these tutors can greatly enhance students' learning (Koedinger, Anderson, Hadley and Mark, 1995).

Although we agree that a more constrained interaction can be beneficial in specific educational settings, for specific types of learners and in particular phases of the learning process, we also believe that supporting more unconstrained, student-led interactions is critical to building richer learning environments that can be beneficial for different types of learners and at different learning stages (VanLehn, Freedman, Jordan, Murray, Osan, Ringenberg, Rose, Shultze, Shelby, Treacy, Weinstein and Wintersgill, 2000; Bunt, Conati, Hugget and Muldner, 2001). We argue that, ideally, the role of a comprehensive model of the student's knowledge and behavior in such environments should be to allow an intelligent coach to dynamically switch from a directive tutoring style to a more open one by taking into account both the student's evolving understanding and her preferred learning style (e.g. autonomous vs. passive). The goal would be to achieve the best tradeoff between a pedagogical interaction that accommodates the student's preferences and one that is more suitable for her evolving level of knowledge.

Our position is supported by empirical studies showing that open learning environments that rely solely on the student's initiative and exploratory capabilities can be highly beneficial for learners with the suitable level of knowledge and learning skills, although they are not as effective for learners who need more structure and guidance in the learning experience (Shute and Glaser, 1990; Bunt et al., 2001). Additional information on when and how more tutorial vs. more exploratory interactions can support better learning (and consequently when a richer student model like Andes' is necessary) could be gained by running user studies that compare the version of Andes presented in this paper with a new, more restrictive version that is currently under development for the U.S. Naval Academy.

7.3 SCALING UP

In order to tutor students through most of a semester, Andes had to have Bayesian networks for around a hundred physics problems. Every time the problem solvers' knowledge base changed,

we had to regenerate all those Bayesian networks. As we discussed in Section 2.1, we were able to automate the network construction process so that it could be done with little human intervention. In particular, we did not have to hand-edit any conditional probability tables. Thus, the knowledge management problem turned out to be no worse for Andes than for any other rule-based knowledge intensive application.

On a few physics problems, updating the Bayesian network using the exact algorithm sometimes took longer than 40 seconds on the 400 MHz Pentium computers that were used in the fall 2000 evaluation. Thus, on some problems, we directed Andes to use stochastic evaluation of the networks and to stop when 40 seconds was reached. On other problems, the update was completely turned off. Hopefully, advances in the Bayesian network update algorithms will allow us to remove these expediencies. Because our networks are generated by a rule-based system, they may have regularities that could be exploited by an improved update algorithm.

On the whole, the use of Bayesian networks for Andes' student model seems to have been a complete success. The design allowed elegant, precise representation of sensible interpretation policies; it did not increase the knowledge management task, and it did not slow the system down too much. Moreover, empirical evaluations of the resulting coaches indicated that students learned more with them than with conventional instruction.

Acknowledgments

This research was sponsored by ONR's Cognitive Science Division under grant N00014-96-1-0260, by ARPA's Computer Aided Education and Training Initiative under grant N660001-95-C-8367, and by AFOSR's Artificial Intelligence Division under grant F49620-96-1-0180.

The authors would like to thank Drs. Robert Shelby, Kay Shulze, Donald Treacy and Mary Wintersgill of the U.S. Naval Academy for their contributions to the development and evaluation of Andes. We also thank Anders Weinstein and Ellen Dugan for helping implement the Andes interface and overall architecture, Marek Druzdzel and Zendong Niu for their contributions to the design and implementation of the student model Bayesian networks, Patricia Albacete for her input at different stages of Andes' design, and Charles Murray for his comments on the manuscript.

References

1. Albacete, P. and VanLehn, K. (2000). The conceptual helper: An intelligent tutoring system for teaching fundamental physics concepts. *Proceedings of Intelligent Tutoring Systems, 5th International Conference, ITS2000,* Montreal, Canada*, Lecture Notes in Computer Science* 1839, Springer, p:564-573.
2. Albrecht, D. W., Zukerman, I. and Nicholson, A. E. (1999). Bayesian Models for Keyhole Plan Recognition in an Adventure Game. *User Modeling and User-Adapted Interaction* 8(1-2), p:5-47.
3. Anderson, J. R., Corbett, A. T., Koedinger, K. R. and Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences* 4(2), p:167-207.
4. Bauer, M. (1995). A Dempster-Shafer approach to modeling agents preferences in plan recognition. *User Modeling and User-Adapted Interaction* 5(3-4), p:317-348.

5. Breese, J., Goldman, R. and Wellman, P. (1994). Introduction to the special section on knowledge-based construction of probabilistic and decision models. *IEEE Transactions on Systems, Man, and Cybernetics*. 24, p:1577-1579.
6. Bunt, A., Conati, C., Hugget, M. and Muldner, K. (2001). On Improving the Effectiveness of Open Learning Environments through Tailored Support for Exploration. *Proceedings of AIED 2001, 10th World Conference of Artificial Intelligence and Education,* San Antonio, TX, U.S.A., p:365-376.
7. Calistri-Yeh, R. J. (1991). An {A*} approach to robust plan recognition for intelligent interfaces. *Applications of Learning and Planning Methods*. Bourbakis, N. G., (Ed.), World Scientific Publishing Co., p:227-251.
8. Carberrry, S. (2001). Techniques for Plan Recognition. *User Modeling and User-Adapted Interaction* 11, p:31-48.
9. Carberry, S. (1990). Incorporating default inferences into plan recognition. *Proceedings of the Eighth National Conference on Artificial Intelligence,* Menlo Park, CA, MIT Press., p:471-478.
10. Charniak, E. and Goldman, R. P. (1992). A Bayesian model of plan recognition. *Artificial Intelligence* 64, p:53-79.
11. Chi, M. T. H. (in press). Self-Explaining: The dual process of generating inferences and repairing mental models. *Advances in Instructional Psychology*.
12. Conati, C. and Carenini, G. (2001). Generating Tailored Examples to Support Learning via Self-explanation. *Proceedings of IJCAI '01, 17th International Joint Conference on Artificial Intelligence,* Seattle, WA, U.S.A, p:1301-1306.
13. Conati, C. and VanLehn, K. (1996a). POLA: A student modeling framework for probabilistic on-line assessment of problem solving performance. *Proceedings of UM-96, 5th International Conference on User Modeling,* Kailua-Kona, Hawaii, U.S.A., User Modeling, Inc., p:75-82.
14. Conati, C. and VanLehn, K. (1996b). Probabilistic plan recognition for cognitive apprenticeship. *Proceedings of the 18th Annual Meeting of the Cognitive Science Society,* San Diego, CA. U.S.A., Erlbaum, p:403-408.
15. Conati, C. and VanLehn, K. (2000). Toward Computer-based Support of Meta-cognitive Skills: A Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education* 11(4), p:389-415.
16. Corbett, A., McLaughlin, M. and Scarpinatto, K. C. (2000). Modeling Student Knowledge: Cognitive Tutors in High School and College. *User Modeling and User-Adapted Interaction* 10(2-3), p:81-108.
17. Corbett, A. T. and Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4(4), p:253-278.
18. Corbett, A. T. and Bhatnagar, A. (1997). Student modeling in the ACT programming tutor: Adjusting a procedural learning model with declarative knowledge. *User Modeling: Proceedings of the Sixth International Conference, UM97,* Chia Laguna, Italy, Springer Wien New York, p:243-254.

19. Dean, T. and Kanazawa, K. (1989). A Model for Reasoning about Persistence and Causation. *Computational Intelligence* 5(3), p:142-150.

20. Gertner, A., Conati, C. and VanLehn, K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model. *Proceedings of the 15th National Conference on Artificial Intelligence,* Madison, Wisconsin, U.S.A., p:106-111.

21. Gertner, A. and VanLehn, K. (2000). Andes: a coached problem solving environment for physics. *Proceedings of Intelligent Tutoring Systems, 5th International Conference, ITS2000,* Montreal, Canada*, Lecture Notes in Computer Science* 1839, Springer, p:131-142.

22. Henrion, M. (1989). Some practical issues in constructing belief networks. *Proceedings of the 3rd Conference on Uncertainty in Artificial Intelligence*, Elsevier Science, p:161-173.

23. Horvitz, E. and Barry, M. (1995). Display of Information for Time-Critical Decision Making. *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence.,* Montreal, Canada, Morgan Kaufmann: San Francisco, p:296-305.

24. Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, R. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* Madison, WI, U.S.A., Morgan Kaufmann: San Francisco., p:256-265.

25. Huber, M. J., Durfee, E. H. and Wellman, M. P. (1994). The automated mapping of plans for plan recognition. *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, p:344-351.

26. Jameson, A. (1996). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction* 5(3-4), p:193-251.

27. Just, M. and Carpenter, P. (1986). *The Psychology of Reading and Language Comprehension*. Boston.

28. Katz, S., Lesgold, A., Eggan, G. and Gordin, M. (1992). Modelling the student in Sherlock II. *Journal of Artificial Intelligence in Education* 3(4), p:495-518.

29. Koedinger, K. and Anderson, J. R. (1993). Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring system design. *Computers as cognitive tools*. Lajoie, S. P. and Derry, S. J., (Ed.). Hillsdale, NJ, Lawrence Erlbaum Associates, p:15-46.

30. Koedinger, K. R., Anderson, J. R., Hadley, W. H. and Mark, M. A. (1995). Intelligent tutoring goes to school in the big city. *Proceedings of the 7th World Conference on Artificial Intelligence and Education,* Charlottesville, NC, AACE, p:421-428.

31. Mahoney, S. M. and Laskey, K. B. (1998). Constructing situation-specific Belief networks. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* S. Francisco, CA, U.S.A, Morgan Kaufmann, p:370-378.

32. Martin, J. and VanLehn, K. (1993). OLAE: Progress toward a multi-activity, Bayesian student modeller. *Artificial Intelligence in Education, 1993: Proceedings of AI-ED 93,* Charlottesville, VA, Association for the Advancement of Computing in Education, p:410-417.

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

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

35. Mislevy, R. (1995). Probability-based inference in cognitive diagnosis. *Cognitive Diagnostic Assessment*. Nichols, P., Chipman, S. and Brennan, R., (Ed.). Hillsdale, NJ., Erlbaum, p:43-71.

36. Mislevy, R. J. and Gitomer, D. H. (1996). The Role of Probability-Based Inference in an Intelligent Tutoring System. *User Modeling and User-Adapted Interaction* 5(3-4), p:253-282.

37. Murray, C. and VanLehn, K. (2000). DT Tutor: A decision-theoretic dynamic approach for optimal selection of tutorial actions. *Proceedings of Intelligent Tutoring Systems, 5th International Conference, ITS2000,* Montreal, Canada*, Lecture Notes in Computer Science* 1839, Springer, p:153-162.

38. Norman, D. A. (1981). Categorization of action slips. *Psychological Review* 88(1), p:1-15.

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

40. Petrushin, V. A., Sinitsa, K. M. and Zherdienko, V. (1995). Probabilistic approach to adaptive student knolwedge assessment: methodology and experiment. *Artificial Intelligence in Education: Proceedings of AI-ED '95,* Washington, DC, U.S.A., p:51-58.

41. Polk, T. A., VanLehn, K. and Kalp, D. (1995). ASPM2: Progress toward the analysis of symbolic parameter models. *Cognitively Diagnostic Assessment*. Nichols, P. D., Chipman, S. F. and Brennan, R. L., (Ed.). Mahwah, NH, Erlbaum, p:127-141.

42. Reggia, J. A. and D'Autrechy, C. L. (1990). Parsimonious covering theory in cognitive diagnosis and adaptive instruction. *Diagnostic Monitoring of Skill and Knowledge Acquisition*. Frederiksen, N., Glaser, R., Lesgold, A. and Shafto, M.G., (Ed.). Hillsdale, NJ, Lawrence Erlbaum Associates, p:191-216.

43. Renkl, A. (1997). Learning from worked-examples: A study on individual differences. *Cognitive Science* 21(1), p:1-30.

44. Reye, J. (1998). Two-phase updating of student models based on dynamic belief networks. *Proceedings of the 4th International Conference on Intelligent Tutoring Systems, ITS '98,* San Antonio, TX, U.S.A.*, Lecture Notes in Computer Science* 1452, Springer, p:274-283.

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

46. Schulze, K. G., Correll, D., Shelby, R. N., Wintersgill, M. C. and Gertner, A. (1998). A CLIPS problem solver for Newtonian physics force problems. *Expert Systems Principles and Programming*. Giarratano, C. and Riley, G., (Ed.). Boston, MA, PWS Publishing Company.

47. Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., Vanlehn, K. and Gertner, A. (2000). Andes: An intelligent tutor for classical physics. *The Journal of Electronic Publishing* 6(1), The University of Michigan Press.

48. Shelby, R. N., Schulze, K. G., Treacy, D. J., Wintersgill, M. C. and Vanlehn, K. (2001). The Andes Intelligent Tutor: an Evaluation. *Proceedings of the Physics Education Research Conference,* Rochester, NY.

49. Shute, V. J. (1995). SMART: Student Modeling Approach for Responsive Tutoring. *User Modeling and User-Adapted Interaction* 5(1), p:1-44.

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

51. Shute, V. J. and Psotka, J. (1996). Intelligent Tutoring Systems: Past, Present and Future. *Handbook of Research on Educational Communications and Technology*. Jonassen, D., (Ed.), Scholastic Publications.

52. Singley, M. K. and Anderson, J. R. (1989). *Transfer of Cognitive Skill*. Cambridge, MA., Harvard University Press.

53. Tulving, E. and Thomson, D. M. (1973). Encoding specificity and retrieval processes in episodic memory. *Psychological Review* 80, p:352-373.

54. Van Mulken, S. (1996). Reasoning about the user's decoding of presentations in an intelligent multimedia presentation system. *Proceedings of UM '96, 5th International Conference on User Modeling,* Kailua Kona, HW, U.S.A, p:67-74.

55. VanLehn, K. (1988). Student modeling. *Foundations of Intelligent Tutoring Systems*. Polson, M. and Richardson, M., (Ed.). Hillsdale, NJ, Lawrence Erlbaum Associates, p:55-78.

56. VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. *Proceedings of Intelligent Tutoring Systems, 3rd International Conference, ITS '96,* Montreal, Canada*, Lecture Notes in Computer Science* 1086, Springer, p:29-47.

57. VanLehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C. P., Shultze, K., Shelby, R., Treacy, D., Weinstein, A. and Wintersgill, M. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. *Proceedings of Intelligent Tutoring Systems, 5th International Conference, ITS2000,* Montreal, CANADA*, Lecture Notes in Computer Science* 1839, Springer, p:474-483.

58. VanLehn, K. and Martin, J. (1998). Evaluation of an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education* 8(2), p:179-221.

59. VanLehn, K. and Niu, Z. (2001). Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. *International Journal of Artificial Intelligence in Education* 12, p:154-184.

60. VanLehn, K., Niu, Z., Siler, S. and Gertner, A. S. (1998). Student modeling from conventional test data: A Bayesian approach without priors. *Proceedings of the 4th International Conference on Intelligent Tutoring Systems, ITS '98,* San Antonio, TX, U.S.A.*, Lecture Notes in Computer Science* 1452, Springer Verlag, p:434-443.

61. Zukerman, I. and Albrecht, D.V. (2001). Predictive Statistical Models for User Modeling. *User-Modeling and User-Adapted Interaction* 11(1-2), p:5-18.

62. Zukerman, I., Albrecht, D. and Nicholson, A. (1999). Predicting Users' Requests on the WWW. *Proceedings of UM '99 , the 7th International Conference on User Modeling,* Banff, Canada., Springer-Verlag., p:275-284.

Authors' Vitae

Cristina Conati

Department of Computer Science, University of British Columbia, Vancouver, BC, V6T1Z4, Canada

Dr. Conati is an Assistant Professor of Computer Science at the University of British Columbia. She received her "Laurea" degree (M.Sc. equivalent) in Computer Science at the University of Milan, and her M.Sc. and Ph.D. in Artificial Intelligence at the University of Pittsburgh. Her research interests include user modeling, reasoning under uncertainty, adaptive interfaces and intelligent learning environments. Her current research focuses on extending the range of user's features that can be captured in a user model - from purely cognitive features (knowledge, goals, preferences), to meta-cognitive skills (such as learning capabilities), personality traits and emotional reactions, in order to widen the spectrum of information that an interactive system can use to adapt its behavior to a user's needs.

Abigail Gertner

The MITRE Corporation, 202 Burlington Road, Bedford, MA, 01730, USA.

Dr. Gertner is a Lead Scientist at The MITRE Corporation, in the Information Technology Center's department of Information Management & Instructional Systems. She received her A.B. in Psychology from Harvard University, and her M.S.E. and Ph.D. in Computer and Information Science from the University of Pennsylvania. Her research interests are primarily in the areas of plan recognition, user modeling, and cooperative response generation in decision support and intelligent tutoring systems. She has pursued these interests in the context of TraumAID, a decision support system for emergency medicine, and Andes, an intelligent tutoring system for university Physics. She is currently developing an ITS to train users of complex software applications.

Kurt VanLehn

Department of Computer Science and Learning and Research Development Center, University of Pittsburgh, Pittsburgh, PA, 15260, USA.

Dr. VanLehn is a Professor of Computer Science, a Senior Scientist at the Learning Research and Development Center, and Director of CIRCLE, an NSF Center. He received a BS from Stanford (Mathematics, 1974) and a PhD from MIT (Computer Science, 1983). Dr. VanLehn's research interests focus on applications of artificial intelligence to education. He is currently engaged in 3 research projects. Circle is an NSF-supported research center that is finding out why human tutoring is so effective and building computer tutors that will equal or even exceed their effectiveness. Circle is a collaborative effort involving 7 professors from CMU and Pitt. The Andes project is developing an intelligent "homework helper" that increases physics student learning without requiring changes in the way the course is taught. The Why2000 project is developing a natural-language based tutoring system for helping students learn how to explain "why" physical systems work the way they do.