Probabilistic Plan Recognition for Cognitive Apprenticeship

Cristina Conati and **Kurt VanLehn**
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA, 15260
conati@pogo.isp.pitt.edu, vanlehn+@pitt.edu

Abstract

Interpreting the student's actions and inferring the student's solution plan during problem solving is one of the main challenges of tutoring based on cognitive apprenticeship, especially in domains with large solution spaces. We present a student modeling framework that performs probabilistic plan recognition by integrating in a Bayesian network knowledge about the available plans and their structure and knowledge about the student's actions and mental state. Besides predictions about the most probable plan followed, the Bayesian network provides probabilistic knowledge tracing, that is assessment of the student's domain knowledge. We show how our student model can be used to tailor scaffolding and fading in cognitive apprenticeship. In particular, we describe how the information in the student model and knowledge about the structure of the available plans can be used to devise heuristics to generate effective hinting strategies when the student needs help.

Introduction

The overall goal of our research is to develop a tutoring system that teaches problem solving skills through cognitive apprenticeship [Collins et al., 1989]. In cognitive apprenticeship the tutor *models* for the students how to solve problems, *scaffolds* students as they first try to solve problems on their own, then gradually *fades* the scaffolding. Scaffolding takes many forms. For instance, students can be led to solve the problem by using an optimal solution and by explicitly performing all the steps in the solution. When this kind of scaffolding has faded out, students can solve problems as they please. In addition, scaffolding requires the tutor to provide help and unsolicited hints when the student is lost. Virtually every pedagogical activity involved in scaffolding faces the difficult problem of interpreting the student's actions. For instance, in order to respond to a student's request for help or to provide unsolicited hints, the coach must determine what line of reasoning the student has been following so that it can construct an appropriate hint.

The problem of inferring from an agent's actions the plan or line of reasoning being followed is known in AI as plan recognition [Kautz and Allen, 1986]. Plan recognition usually involves inherent uncertainty [Carberry, 1990, Charniak and Goldman, 1993, Huber et al., 1994] and in cognitive apprenticeship it is an especially hard problem [Self, 1988], since cognitive apprenticeships teach intellectual skills where most of the important activity is hidden from the coaches' view. In this paper we describe an evolving student modeling framework, based on [Conati and Vanlehn, 1996, Martin and VanLehn, 1995], that performs plan recognition by integrating probabilistic reasoning and information on the student's mental state with knowledge of available plans.

Very little research has been devoted to plan recognition in student modeling, none of which includes probabilistic plan recognition. Probabilistic reasoning has been applied in student modeling only to perform *knowledge tracing* [Anderson et al., 1995], that is to assess the student's domain knowledge and possible misconceptions from problem solving performance [Corbett et al., 1995, Martin and VanLehn, 1995, Mislevy, 1995, Petrushin, 1993]. Most of the attempts to apply plan recognition to intelligent tutoring systems rely only on the library of the available plans, without taking into consideration the student's degree of mastery in the target domain assessed by knowledge tracing to help discriminate among alternative interpretations of the students' actions [Genesereth, 1982, Kohen and Greer, 1993, Ross and Lewis, 1988]. The Andersonian tutors [Anderson et al., 1995] perform both knowledge tracing and model tracing, a very simple form of plan recognition, but they do not integrate the two kinds of assessment. Instead, they reduce the complexity of plan recognition by restricting the number of acceptable solutions that the student can follow and by asking the student when there is still ambiguity among two or more solutions.

In the first part of the paper we describe how our student model uses a compact, graph-based representation of plans that encodes the plausible lines of reasoning for solving a problem in the target task domain (Newtonian physics in this application) and how the graph and the student's actions are used to dynamically generate a Bayesian network [Pearl, 1988] that performs plan recognition and knowledge tracing. In the second part of the paper we describe how a tutoring system based on cognitive apprenticeship can use the student model to tailor scaffolding and fading. In particular we describe how we integrate probabilistic plan recognition, knowledge tracing and hint selection rules to generate effective hints when a student needs help.

A graph based model of the problem solving process

In order to decide which line of reasoning underlies a student's action the system must have a set of lines of reasoning that students may pursue. This set represents the solution space of a problem and in domains like physics can be quite large. The data structure that we use to represent the solution space of a problem is the *solution graph*. The solution graph is automatically built from a knowledge base of production

"BOY+FLOUR" PROBLEM
A boy having a mass of 75kg (Mb), holds in his hands a bag of flour weighing 40N (Wf). With what force N does the floor push up on the boy's feet?

SOLUTION PLAN "SummWeight"	SOLUTION PLAN "SummMass"
Mb=75;	Mb=75;
Wf = 40;	Wf = 40;
G = 9.8;	G = 9.8;
Wb = Mb*G;	Mf = Wf/G;
Wa = Wf + Wb;	Ma = Mf + Mb;
N = Wa;	Wa = Ma*G;
	N = Wa;

(Wa = total weight of boy and flour; Ma = total mass of boy and flour)

Figure 1: A simple problem and its correct solution plans

rules that contains the physics principles necessary to solve Newtonian physics problems. It can compactly represent the solution space for any given physics problem.

The solution graph contains three types of information, which represent (1) all the plans to solve the problem that can be derived by the rules in the physics knowledge base; (2) all the algebraic solution paths that develop these plans; (3) the reasoning behind each step in a plan. Let's consider, for example, the solution graph for the physics problem in Figure 1. There are two different plans for this problem: either find the weight of the flour and add the weights or find the mass of the boy, add the masses, then convert them to weight. Then equate the total weight of the boy and the flour to the sought normal force. These two plans are represented by the two sets of primitive equations "SummWeight" and "SummMass" in Figure 1. We define as primitive equations those equations that are direct applications of physical laws or mathematical principles, or quantities given in the problem statement.

The primitive equations for a plan can be generated and combined in many different ways, generating a large number of solution paths. Solution paths can be generated, for example, by forward chaining, backward chaining, depth first, breadth first or any combination of these strategies. Existing tutoring systems that provide support during problem solving reduce the number of the acceptable solution paths by forcing the students to follow a particular problem solving strategy [Anderson et al., 1995, Derry and Hawkes, 1993]. Our solution graph, on the other hand, provides a compact representation of all the possible solution paths that develop a given plan and supports tutorial interaction that is more flexible and similar to those generated by human physics tutors.

A simplified solution graph for the problem in Figure 1 is represented in Figure 2A. It contains nodes representing primitive equations and problem variables (ellipses and diamonds respectively in Figure 2A). Primitive equations correspond to the application of rules of the knowledge base that encode quantitative physics principles. When a rule is applied an *application node* representing the corresponding primitive equation is entered in the solution graph, along with its parent nodes representing the known variables and its child node representing the computed variable. Given the solution graph, each equation entered by the user is interpreted by decomposing it into primitive equations and by marking the corresponding application nodes in the solution graph, as shown in Figure 2A.

Behind each application node stands a dependency network that records the derivation of the corresponding primitive equation. For instance, a simplified dependency network for the equation $N = Wa$ is shown in Figure 3. There is one dependency network for each application node, but the dependency networks of different nodes often share large sub-nets. For simplicity, in the solution graph of Figure 2A the complete dependency networks are represented by single physics rules, the rectangles in the graph. The dependency networks are needed in order to interpret actions that are not equations. For instance, suppose the student initializes a force diagram for the compound object consisting of the boy and the bag. This action corresponds to the darkened node labelled "boy-bag is an object" in Figure 3.

The solution graph allows the tutor to accept the student's actions in any order, as long as they belong to a known plan. In fact any traversal of the graph that connects soughts to givens represents a legal solution path within a specific plan, and vice versa every correct solution path corresponds to a traversal of the solution graph. The solution graph also allows to keep track of the actions that the student's has performed so far.

Bayesian interpretation of student actions

After a student's action has been mapped on the solution graph, the system uses the mapping and the structural information encoded in the solution graph to update a Bayesian network in charge of plan recognition. A Bayesian network [Pearl, 1988] is a directed acyclic graph where nodes represent random variables and arcs represent probabilistic dependencies among the variables. In our Bayesian network, the random variables represent pieces of domain knowledge, possible plans to solve a problem, the student's actions and the possible inferences that might have generated the actions.

Figure 2B shows the state of the Bayesian network after it has been incrementally updated with the two equations $N = Wa$ and $Mf = 40/G$. The nodes at the bottom of the network, called *action nodes*, represent the entered equations. For each action node, there is a *derivation node* for any way found in the graph to derive the equation (nodes *der1* and *der2* in Figure 2B) and an additional derivation node that represents any other way in which the action could have been derived, for example by guessing or by copying from a previous problem (nodes *other1* and *other2* in Figure 2B). The two equations in our example can be derived in only one way, therefore only one derivation node (besides the *other* node) is inserted for each of them. If the student typed the equation $Wa = 775$ the system would find in the graph two ways to generate it, and two derivation nodes would be inserted in the network. Both action and derivation nodes have values TRUE/FALSE representing the probability that the actions or the derivations have (or have not) been performed.

An action node is linked to the corresponding derivation nodes via an OR link matrix, which defines the conditional probability distribution of the action node given the probability distribution of its parents as a logical OR. That is, an action node is TRUE if at least one of its parent derivation nodes is TRUE and FALSE otherwise. Each derivation node is linked through an AND link matrix to its parent nodes, correspond-

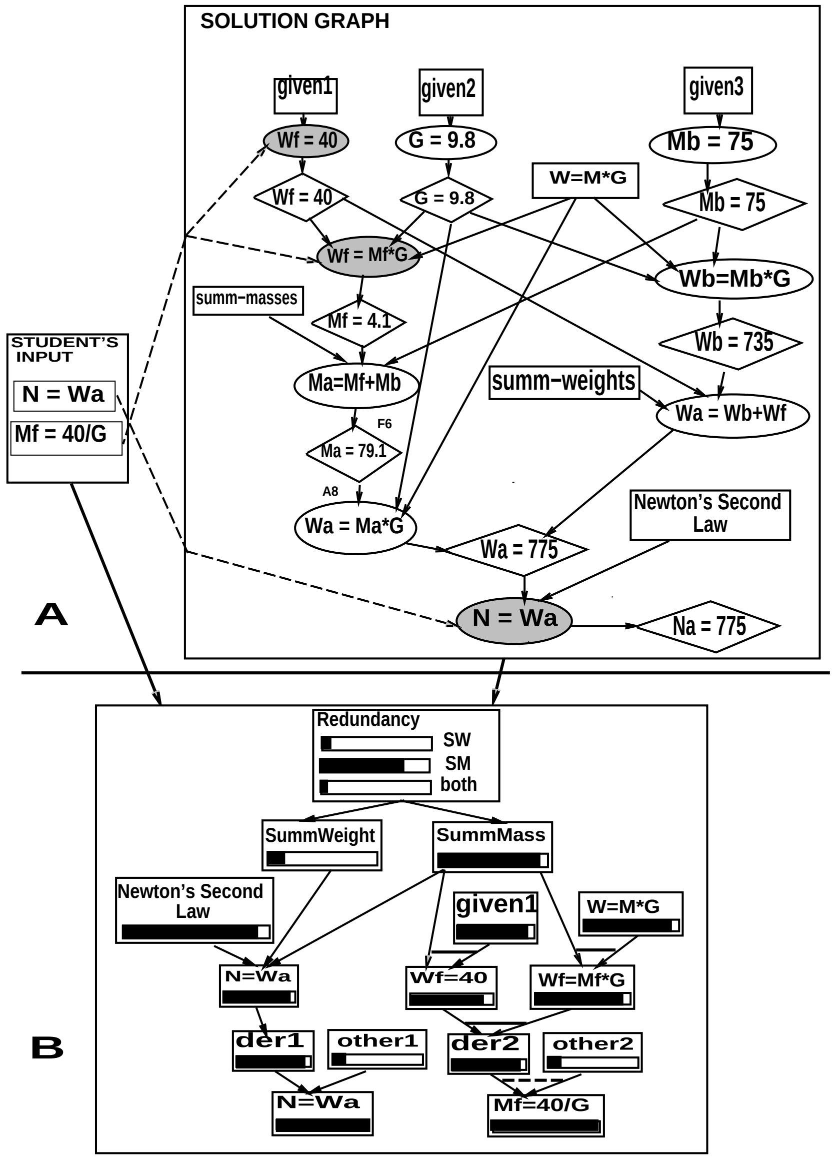

Figure 2: Example of solution graph and Bayesian network for the problem in Figure 1

ing to the application nodes marked in the solution graph as generating the derivation. In Figure 2B, for example, the parent application nodes of the derivation node *der2* are the application nodes labeled $Wf = 40$ and $Wf = Mf*G$. The AND link matrix between derivation and application nodes represent the fact that a derivation occurs if and only if all the necessary rules and givens have been applied.

Each application node is linked to a node that represents the corresponding rule and to a node representing the plan to which the rule application belongs. The solution graph indicates to which plan each application node belongs and the corresponding links are inserted in the Bayesian network. The probability of the TRUE value for a plan node represents the probability that the student is following that plan. The probability of the TRUE value for a rule node represents the probability that the student knows that rule. The link matrix between an application node and its parents is a leaky-AND, to represent the fact that when a student generates a rule application she almost always knows the parent rule and is following the parent plan, although there is a small probability that she generated the application without actually knowing the rule or without having that plan in mind.

All the plan nodes in the Bayesian network are linked to a common ancestor, the node labeled as *redundancy* in figure figure 2B. The values of this node allows to explicitly represent the probability that the student is following only one of the available plans (values SW and SM in figure 2B) or more then one plan at a time (value $both$ in figure 2B).

The action nodes at the bottom of the network represent evidence coming from the student, therefore their TRUE value is clamped to 1 and that evidence is propagated upward in the network via a Bayesian update algorithm (we use the Lauritzen-Spiegelhalter algorithm [Pearl, 1988]). In Figure 2B the higher probability of the "SummMass" node reflects

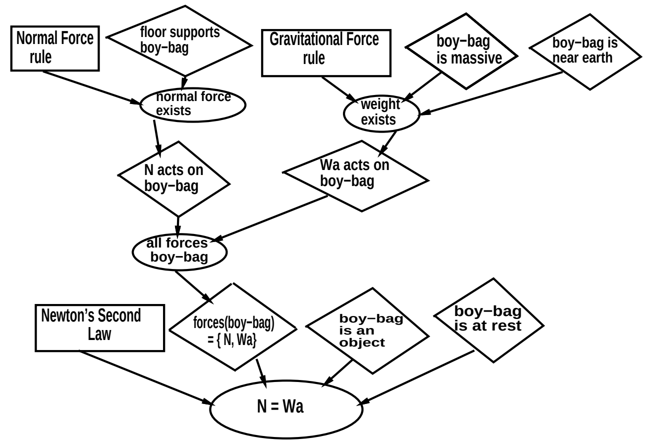

Figure 3: Dependency network for the application node "N=Wa"

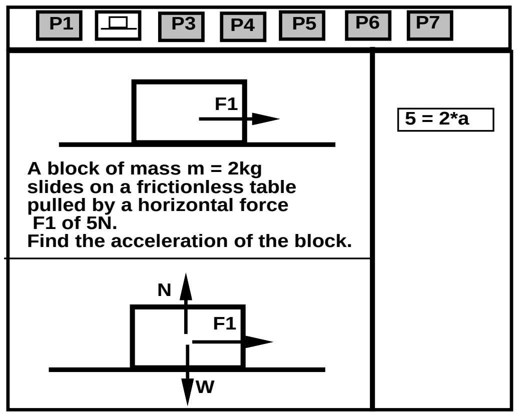
P1
P3
P4
P5
P6
P7
F1
5 = 2*a
A block of mass m = 2kg slides on a frictionless table pulled by a horizontal force F1 of 5N. Find the acceleration of the block.
N
F1
W

Figure 4: Example of ambiguous student's action

the fact that, although the equation $N = Wa$ belongs to both solutions, the equation $Mf = 40/G$ provides clear evidence for the plan "SummMass". In addition, the propagation of evidence to rule nodes in the network provides assessment on the student's knowledge of the corresponding physics principles. After the student has finished with the problem, the probabilities of the rule nodes and the dependencies between rules are read out of the Bayesian network and become the updated student model [Martin and VanLehn, 1995] that will be used to aid plan recognition in future interactions with the student, as we will see in the next section.

Using the student model to tailor the scaffolding

One form of scaffolding in cognitive apprenticeship consists of forcing the student to explicitly perform all the steps in a solution, for example to draw all the forces involved in a physics problem before entering equations. This kind of scaffolding, called *reification*, is usually applied to novice students and faded when the students become more skilled in the target domain. Fading makes the interpretation of the student's actions more difficult, because more of the student reasoning is hidden from the tutor. Our student model allows the system to handle the increased ambiguity generated by fading and to make more flexible decisions about when to fade the scaffolding, instead of relying on fixed rules such as "fade reification after the student has solved 5 problems correctly". Let's suppose, for example, that after solving the problem in the previous section with reification turned on our student starts solving the problem in the upper left window in Figure 4. The student types the equation in the right window in Figure 4, without drawing forces in the bottom left window. Should the tutor keep reification turned on and ask the student to draw the forces? The tutor uses the probabilities in the Bayesian network generated after the student's action to make the decision.

The equation that the student typed can be generated either by the correct version of Newton's Second Law, $F = ma$, in which F is the sum of all the three forces acting on the body (shown by the vectors in the bottom left window) or by an incorrect version in which F is any force applied to the body. If this incorrect version of Newton's Second Law is included in the system's knowledge base, a corresponding incorrect plan will be included in the problem's solution graph and the Bayesian network generated after the equation $5 = 2 * a$ will be the one in Figure 5. If the only information available to the system was the entered equation, then propagation of this evidence in the network would assign equal probability to the two possible derivations and to the two possible plans and the only way for the tutor to understand if the student followed the correct or the incorrect plan would be to force the student to draw the forces in the free body diagram. However, since the student model generated by the previous interaction with the student reports a high probability that the student knows all the pieces of knowledge required to correctly apply Newton's Second Law (the rule nodes in Figure 3), then the Bayesian network assigns a high probability to the correct derivation and to the correct plan and the tutor can avoid imposing the reification of forces.

Another fundamental component of scaffolding is the capability to provide help. Providing help during problem solving is a delicate pedagogical problem. A request for help indicates that the student reached an impasse in the problem solving process. The impasse can be turned into a learning episode if the tutor helps the student generate the inferences necessary to fill the knowledge gap that created the impasse. Hinting is one of the strategies often used by human tutors to provide constructive help [Hume et al., 1996]. Of course, the necessary condition to provide useful hints is that the tutor understands why the impasse happened. Sometimes the student can articulate for the tutor what her problem is, but often the student is too confused to be able to tell why she is unable to continue and the tutor must use alternative criteria to decide what is the best way to help the student solving the impasse.

The structural information in the solution graph and the probabilities generated by the Bayesian network can be used to devise strategies for selecting what to hint for (i.e the hint target). The Bayesian network generates predictions about what plan the student is following. Given the plan, a hint generation module determines from the solution graph what steps have already been performed along this plan and what steps are left. The steps left to be performed become the *hinting set* within which the hint generator chooses the target

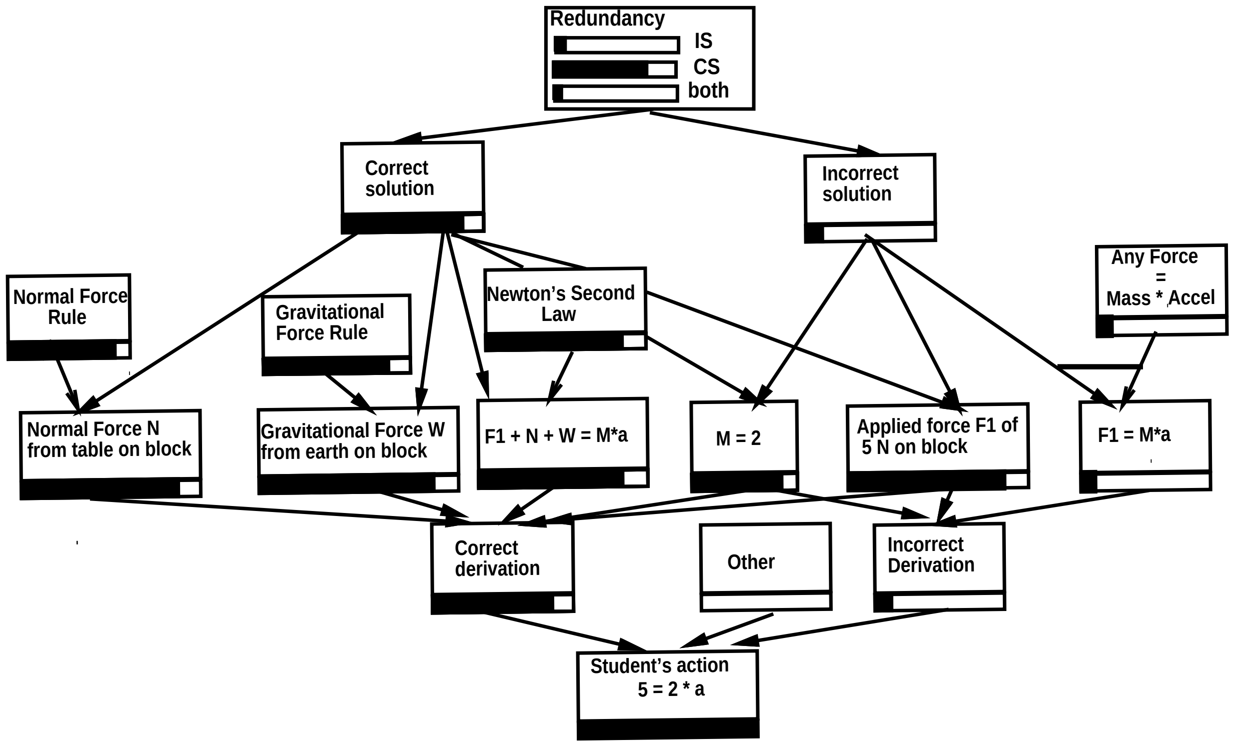

Figure 5: Bayesian network built after the student's action in Figure 3

of the next hint. Let's suppose, for example, that the student asks for help after typing the two equations in Figure 2A. At this point *SummMass* is the most probable plan, as shown in the network in Figure 2B, and the hinting set consists of the following primitive equations:

$\{G = 9.8, Mb = 75, Ma = Mb + Mf, Wa = Ma * G\}$

Several heuristics can be used to select among equations in the hinting set. One heuristic gives minimal priority to hints that simply remind the student of values that are problem givens. In our example this heuristic would rule out using the steps $G = 9.8$ and $Mb = 75$ as hints. Two different heuristics can be based on the probabilistic assessment of the student's knowledge of physics rules. The *reminding* heuristic selects steps related to physics knowledge that has high probability of being mastered. The hints generated by this heuristic are analogous to those categorized in [Hume et al., 1996] as *pointing hints*, that is hints pointing to well-known information that the student doesn't realize is relevant in the current situation. In our example this heuristic would select the step $Wa = Ma*G$, since it is related to the rule $W = M * G$ (see solution graph Figure 2A) which has reached a high probability of being known (see Bayesian network in Figure 2B) after the student typed $Mf = 40/G$.

The second heuristic, the *low knowledge* heuristic, selects for hinting a step related to physics knowledge that is not evidently known by the student. This heuristic generates hints analogous to those classified in [Hume et al., 1996] as *conveying information* hints, that suggest information prompting the student to infer an answer or the next step of a solution. In our example, this heuristic would select for hinting the step $Ma = Mb + Mf$, since there is no direct evidence from the student's actions that the student knows the related rule "the total mass of a system is the sum of the masses of the system's components".

A third heuristic that can be combined with either of the two above is the *adjacency heuristic*. It selects a step adjacent in the solution graph to the last step performed by the student. In our example, the only step adjacent to the last performed, $Mf = 40/G$, is $Ma = Mb + Mf$. How to combine these heuristics and how to set their priorities when they provide conflicting suggestions is an open research issue, since not many results are available about how human tutors select their hints and how effective they are [Hume et al., 1996]. One of our hypotheses is that the adjacency heuristic should have high priority since it is important that the content of the hint preserves a discernible connection with what the student is trying to do. A second hypothesis is that the *reminding* heuristic should be used with care, to avoid suggesting to the student something that she already knows how to perform. To this regard, a variation of the *reminding heuristic* would be to consider whether the high probability of the target knowledge derives from recent student actions (as in our example) or from actions that the student has performed in previous problems. If the target knowledge has been used recently, it is less likely that the student needs to be reminded of it. For instance, in our example the student has just applied the rule $W = M * G$ to generate $Mf = 40/G$, therefore it is not very plausible that now she is having problems in applying the rule to find Wa. Given this variation the *reminding heuristic* all the three heuristics listed above point to the selection of $Ma = Mf + Mb$ as the hint target. We are planning to test these and additional/alternative hypotheses and heuristics by using them to implement different hint selection strategies and by testing these strategies with real students.

Conclusions

In this paper we have described a student modeling framework that performs knowledge tracing and plan recognition while students solve problems in Newtonian physics. Our approach to plan recognition is innovative in that we use a Bayesian Network to integrate in a principled way knowledge about the student's behavior and mental state with knowledge about the available plans. The available plans are encoded in a graph-based representation that compactly represents all the different orders in which each plan can be implemented and allows maximum flexibility when accepting the students' solutions. We have presented examples of how our student model can be used to tailor cognitive apprenticeship, in particular to generate effective hints when the student needs help.

We have started to evaluate the accuracy of the predictions generated by our model on the solutions generated by three students solving the problem presented in this paper and on the solutions generated by two students solving a more complex problem involving 4 different solution plans and 18 primitive equations. The predictions generated by the model after each step of the 5 solutions have been consistent with the plan that each student actually followed. We plan to perform a more extensive evaluation after we have formalized the hinting rules and reimplemented a more efficient C++ version of the update algorithm for the Bayesian network, currently implemented in Lisp. The results of the evaluation will decide whether we will need to switch to approximate update algorithms[Cousins et al., 1993] to maintain acceptable performances on more complex problems.

Acknowledgements

This research is supported by ONR's Cognitive Science Division under grants N00014-96-1-0260 and N00014-95-1-0950, and by DARPA's Computer Aided Education and Training Initiative under grant N66001-95-C-8367.

References

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

[Carberry, 1990] Carberry, S. (1990). Incorporating default inferences into plan recognition. In *Proceedings of AAAI-90*, pages 471–478.

[Charniak and Goldman, 1993] Charniak, E. and Goldman, R. (1993). A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79.

[Collins et al., 1989] Collins, A., Brown, J. S., and Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In Resnick, L. B., editor, *Knowing, learning and instruction: Essays in honor of Robert Glaser*, pages 543–494. LEA, Hillsdale, NJ.

[Conati and Vanlehn, 1996] Conati, C. and Vanlehn, K. (1996). Pola: a student modeling framework for probabilistic on-line assessment of problem solving performance. In *Proc. of UM-96, 5th International Conference on User Modeling*.

[Corbett et al., 1995] Corbett, A. T., Anderson, J. R., and O'Brien, A. T. (1995). Student modeling in the ACT programming tutor. In Nichols, P. D., Chipman, S. F., and Brennan, R. L., editors, *Cognitively Diagnostic Assessment*, pages 19–42. LEA, Hillsdale, NJ.

[Cousins et al., 1993] Cousins, S., Chen, W., and Frisse, M. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine*, 5:315–340.

[Derry and Hawkes, 1993] Derry, J. S. and Hawkes, L. W. (1993). Local cognitive modeling of problem-solving behavior: An application of fuzzy theory. In Lajoie, S. and Derry, S., editors, *Computers as Cognitive Tools*. Lawrence Erlbaum, Hillsdale, NJ.

[Genesereth, 1982] Genesereth, M. (1982). The role of plans in intelligent teaching systems. In Sleeman, D., editor, *Intelligent tutoring systems*, pages 137–156. Academic Press, New York.

[Huber et al., 1994] Huber, M., Durfee, E., and Wellman, M. (1994). The automated mapping of plans for plan recognition. In *Proceedings of the tenth conference on Uncertainty in Artificial Intelligence*, pages 344–351.

[Hume et al., 1996] Hume, G., Michael, J., and Evens, M. (1996). Hinting as a tactic in one-to-one tutoring. *The Journal of Learning Sciences*, 5(1):23–47.

[Kautz and Allen, 1986] Kautz, H. and Allen, J. (1986). Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37.

[Kohen and Greer, 1993] Kohen, G. and Greer, J. (1993). Recognizing plans in instructional systems using granularity. In *Proceedings of the 4th International Conference on User Modeling*, pages 133–138.

[Martin and VanLehn, 1995] Martin, J. and VanLehn, K. (1995). A bayesian approach to cognitive assessment. In Nichols, P., Chipman, S., and Brennan, R. L., editors, *Cognitively diagnostic assessment*. LEA, Hillsdale, NJ.

[Mislevy, 1995] Mislevy, R. J. (1995). Probability-based inference in cognitive diagnosis. In Nichols, P., Chipman, S., and Brennan, R., L., editors, *Cognitively diagnostic assessment*. LEA, Hillsdale, NJ.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible inference.* Morgan Kaufmann, Los Altos, CA.

[Petrushin, 1993] Petrushin, V. A.and Sinitsa, K. M. (1993). Using probabilistic reasoning techniques for learner modeling. In *Proceedings of the 1993 World Conference on AI and Education*, pages 426–432.

[Ross and Lewis, 1988] Ross, P. and Lewis, J. (1988). Plan recognition for intelligent tutoring systems. In Ercoli, P. and Lewis, R., editors, *Artificial Intelligence tools in Education*, pages 29–37. Elsevier Science Publishers.

[Self, 1988] Self, J. (1988). Bypassing the intractable problem of student modeling. In *Proc. of ITS-88*, pages 18–24, Montreal, Canada.