POLA: a student modeling framework for Probabilistic On-Line Assessment of problem solving performance

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

Abstract

The paper presents POLA, a student modeling framework that performs probabilistic assessment of students' performance while they solve problems in introductory Physics. Existing efforts toward probabilistic student modeling focus exclusively on performing knowledge tracing. With POLA we aim to turn OLAE, a system that performs probabilistic knowledge tracing, into a system that applies probabilistic reasoning to perform both knowledge and model tracing. POLA generates probabilistic predictions about the student's line of reasoning without using heuristics, even when the problem's solution space is large. An AND/OR graph provides a compact representation of all the available solutions for a problem. A Bayesian network built incrementally from the AND/OR graph and from the student's actions generates predictions about the solution that the student is following. At the end of the problem solving session the network provides an assessment of student's level of mastery of the physics knowledge involved in the problem's solution.

Introduction

There has been a great deal of interest recently in applying probabilistic methods, especially Bayesian networks, to reasoning about uncertainty (Pearl 1988). The problem of inferring from a tutorial interaction what a student is thinking and what knowledge she has clearly is a problem of uncertain reasoning, so there have been several recent attempts to apply probabilistic reasoning to student modeling (Villano 1992; Martin & VanLehn 1995; Petrushin 1993; Sime 1993; Mislevy 1995; Duncan, Brna, & Morss 1994; Gitomer *et al.* 1995). This paper discusses a new approach to applying Bayesian networks to student modeling.

There are actually two types of student modeling, which Anderson, Corbett and Koedinger (Anderson *et al.* 1995) called *knowledge tracing* and *model tracing*. These are their names for the particular techniques they use, but the distinction is perfectly general. Knowledge tracing refers to the problem of determining what students know, including both correct domain knowledge and robust misconceptions. Model tracing refers to tracking a student's problem solving as she works on a problem.

Model tracing is useful for systems that attempt to answer requests for help or to give unsolicited hints and feedback in the middle of problem solving. In fact, to do an adequate job of helping, hinting and critiquing an on-going solution attempt a system must at minimum understand what line of reasoning the student is attempting to pursue.

On the other hand, knowledge tracing is useful for making longer range pedagogical decisions, such as what problem to assign next or what evaluation grade to assign to the student. All the existing attempts to apply probabilistic reasoning to student modeling have been directed toward performing knowledge tracing only.

The research project reported in this paper aims to turn an existing system that does probabilistic knowledge tracing, OLAE (Martin & VanLehn 1995), into a system that applies probabilistic reasoning to do both knowledge and model tracing. The new system is called POLA (Probabilistic On-Line Assessment). It will eventually become the student modeling component of a tutoring system that offers help upon request and gives unsolicited hints and feedback during problem solving. The challenge is to get POLA to follow the student's reasoning without making unwarranted heuristic assumptions. Instead, uncertainty in the interpretation of the student's actions is to be handled in a sound probabilistic manner. In particular, whenever multiple solutions paths are consistent with the student's past actions, POLA will be able to assess which one is most probably the one that the student is following. Given such information, the tutoring system can generate reasonable hints and answer help requests. Without it, the tutoring system would be unable to answer even the simplest help request, such as "what should I do next?"

Student modeling in domains with large solution spaces

Both the OLAE research and this research have been conducted in the task domain of physics (Newtonian mechanics in particular). Physics is somewhat different from other task domains in the student modeling literature in that there are many correct solution paths. For instance, for the problem shown in Figure 1a the 8 primitive equations in (Figure 1b) can be grouped in the two *solution sets* in Figure 1c, corresponding to two conceptually different solutions, $SummWeight$ and $SummMass$. Since equations can be generated in any order, each solution can be generated in at least $n!$ different ways, where n is the number of

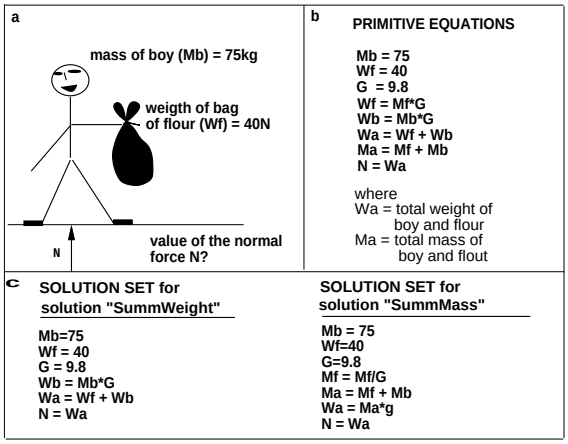

Figure 1: Physics problem with multiple solutions

primitive equations in the corresponding solution set. Thus, there are $6! = 720$ ways to express $SummWeight$ and $7! = 5040$ ways to express $SummMass$. Actually these numbers represent a lower bound for the number of different ways of expressing solutions for this problem because they assume that the student only writes primitive equations. When combinations of primitive equations are written, such as $Wa = 40 + Mb * 9.8$ the number of correct solution paths increases dramatically.

Other task domains that, like physics, are characterized by large solution spaces are, for example, algebra word problems, geometry theorem proving and programming. One approach to handle the complexity of model tracing in such domains is to limit the number acceptable solution paths and to make the student follow one of the acceptable paths (Corbett & Anderson 1992; Singley 1990; Derry & Hawkes 1993). However, the approach of limiting the acceptable solutions to a predefined subset can be too constraining for the student. Extended studies on the educational strategies employed by human tutors (Merril, Reiser, & Ranney 1992) revealed that teachers usually let the student do as much of the work as possible during problem solving, allowing the student to maintain a feeling of control and intervening only when the student makes mistakes. Therefore, POLA aims to provide a student modeling framework that is able to perform model tracing in domains with vast solution spaces without imposing unwarranted limitations on the set of acceptable solutions.

The starting point: OLAE

OLAE is an off-line probabilistic assessment tool that collects data from students solving problems in introductory college physics. It provides an interface that presents problems and allows the student to solve them by entering algebraic equations. OLAE uses the student's actions to assess the probability that the student knows the physics rules encoded in its model of physics knowledge. The assessment is performed off-line by propagating the evidence provided by the student's actions into a Bayesian network built upon

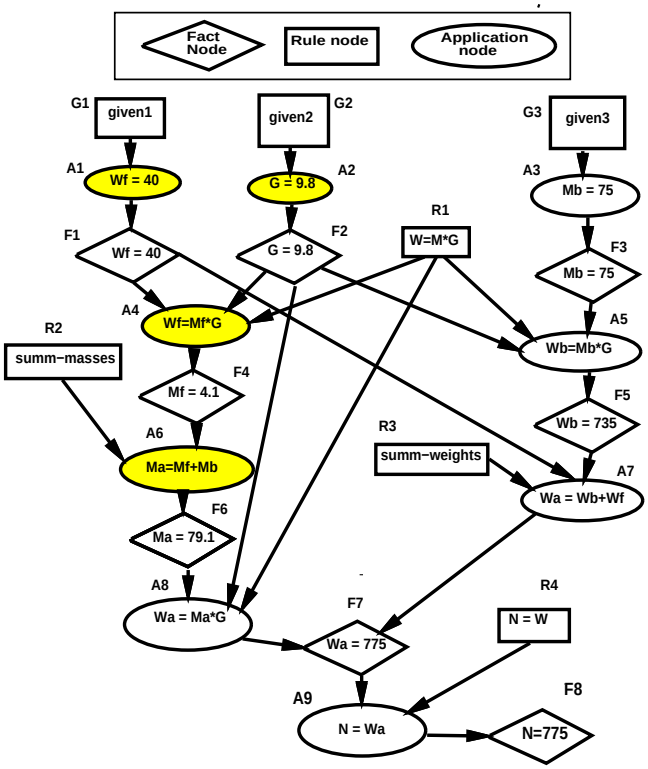

Figure 2: Solution graph for the problem in figure 1

an AND/OR graph representation of the problem solution.

The AND/OR representation of problem solutions

The AND/OR graph is built automatically by a problem solver from a knowledge base of production rules that encodes the physics knowledge necessary to solve problems in Newtonian physics. Some of the rules in the knowledge base encode quantitative physics principles that must be applied to solve the problem (such as $W = M * G$). Others encode more qualitative knowledge such as "a tension force exists when a body is tied to a string." Figure 2 shows a simplified version of the AND/OR graph for the problem in Figure 1, in which only nodes related to quantitative derivations are represented.

The problem solutions are generated forward chaining by the problem solver starting from the problem givens and firing rules as soon as they become applicable. Three kinds of nodes are present in the AND/OR graph. (1) *Rule nodes* (rectangles in Figure 2) that represent rules and problem givens. (2) *Application nodes* (ellipses in Figure 2) that explicitly represent rules firing. (3) *Fact nodes* (diamonds in Figure 2), that represent conclusions derived during problem solving. Application nodes are the AND nodes in the graph, since for a rule to be applied to generate a conclusion from certain antecedents the rule and all the antecedents must be known. Each application node is connected to the fact node representing the derived conclusion. Fact nodes are OR nodes in the graph, modeling the fact that some conclusions

can be derived in multiple ways (see, for example, node F7 Figure 2). At the end of the problem solving process the system has generated an AND/OR graph that encodes all the conceptually different ways in which rules and given data can be combined to generate intermediate and final results for the problem.

Along with the AND/OR graph the system generates for each problem a database in which the equations corresponding to facts in the AND/OR graph are stored in a normalized form independent from the algebraic form in which they are written. For example, the equations $Mf = 40/9.8$ and $40 = Mf * 9.8$ have the same normalized form and would correspond to the same entry in the database, indexed to node F4 in Figure 2. The database allows OLAE to quickly verify the correctness of a student's equation by normalizing it and then by matching it with the database entries.

Probabilistic assessment with OLAE

OLAE assesses the student knowledge of physics rules from the student's solution of a problem using a Bayesian network consisting of the AND/OR graph for that problem integrated with nodes that represent the equations entered by the student, called *action nodes*. Each node in the Bayesian network has values TRUE/FALSE, which represent (a) for a rule node, whether the student knows the corresponding rule. (b) For an application node, whether the student actually used a rule during the problem solution. (c) For a fact node, whether the student knows the associated fact about the given problem. (d) For an action node, whether the student has performed the action.

The conditional probabilities used in the Bayesian network are derived from two assumptions. First, OLAE assumes that rule application is almost logical. Namely, if the rules and all its antecedents are known, then the rule will "almost always" be applied, where "almost always" is represented by a leaky-AND link matrix which encodes the probability that the AND boolean function is computed incorrectly. Second, OLAE assumes that people rarely infer the same fact twice and encodes this relationship with a leaky-XOR link matrix between each action node and its parent derivation nodes.

OLAE starts to assesses the student's knowledge of physics rules when the student has finished entering her solution. Each equation in the student's solution is translated into normalized form and looked up in the equation database. If a match is found, the equation represents a conclusion in the AND/OR graph and the associated fact node is retrieved. A new action node representing the entered equation is linked to the corresponding fact node. The new action node is clamped to a probability of 1.0 and the new evidence is propagated through the network. When all the actions performed by the student have been inserted into the network the probabilities associated with the rule nodes provide a prediction of the student knowledge of the corresponding physics principles.

From OLAE to POLA

The aim of OLAE is to perform knowledge tracing, not to provide support during problem solving. Therefore, OLAE does not need to monitor the student's performance on-line or to predict the student's line of reasoning. Two major changes have been necessary to adapt the OLAE architecture to provide POLA with the capability to perform model tracing in addition to knowledge tracing.

Exploiting the AND/OR graph

The first change that we made to OLAE was to extend the use of the AND/OR graph to keep track of the progression of the student in the solution space.

In order to monitor the student's progression POLA must be able to infer from any correct equation that the student types the rules that have been used and the facts to which they have been applied to derive the equation. This is simple if the equation represents a conclusion along a problem solution, such as $Ma = 79.1$. In fact, to find all the rules and conclusions that have been used to generate the equation it is enough to retrieve in the AND/OR graph the fact node corresponding to the equation (node $F6$ in Figure 2) and all its ancestors nodes.

The matter is more complicated for equations that combine multiple but incomplete rule applications, such as $Ma = 4.1 + Mb$, since they are not directly associated with nodes in the AND/OR graph. From $Ma = 4.1 + Mb$, POLA must be able to infer that (a) Rule *W=M*G*, represented by node R1 in Figure 2, has been applied to the values of Wf and G to generate the value of 4.1 for Mf; (b) Rule *summ-masses*, represented by node R2 Figure 2, has been applied to the computed value of Mf and to the variable Mb, whose value has not has not been substituted yet.

A possible approach to this problem is to force the student to write every rule application explicitly[1]. To avoid imposing this constraint on the student, we have developed an algorithm that infers from any correct equation that the student types the rules and the givens that have been employed to derive it. Every time the student enters an equation that belongs to one of the problem's solutions POLA computes the rule applications entailed in the equation and marks the corresponding application nodes in the AND/OR graph. The shaded nodes in Figure 2, for instance, are marked in the AND/OR graph if the student types $Ma = 4.1 + Mb$.

It is important to emphasize that, although the AND/OR graph is generated by forward chaining, solution steps can be generated by the student in any order. In summary, the AND/OR graph provides a compact representation of all the solution sets and solution paths for a problem and it allows POLA to keep track of the actions that the student's has performed.

A new topology for the Bayesian network

The second change we made to OLAE to add the capability to perform model tracing consisted in developing a new

[1]This is the strategy that OLAE adopts

Bayesian network capable of assessing, for each solution set in a problem, the probability that the student is following it given the student's actions.

We initially thought that we could perform probabilistic model tracing using the OLAE Bayesian network itself. However, we soon found that propagation through this network caused incorrect assignment of probabilities. In particular, evidence in a Bayesian network is propagated through every link, not only from a clamped node backward to its parents. Therefore, adding action nodes directly to the AND/OR graph, as was done in OLAE, has the additional effect of propagating evidence forward to nodes derived from the facts related to the performed actions. For instance, if the student enters the equation $Ma = 79.1$ a corresponding action node is attached to the fact node F6 in Figure 2. Part of the effect of the evidence provided by the action $Ma = 79.1$ is to increase the probability of fact F6, rule R1 and fact F1. This consequently makes both application node A8 and fact node F7 very probable. The meaning of high probabilities for these two nodes is very different from the meaning of high probabilities for nodes that are ancestors of the performed action. While high probabilities for the ancestor nodes mean that the information encoded by these nodes has been correctly used to derive the action, the only possible meaning for high probabilities of descendant nodes such as A8 and F7 is that most likely they will be used in or derived by successive student actions.

A possible revision of the OLAE Bayesian network is to capture these different meanings by adding more values to nodes in the network, such as "ALREADY APPLIED", "ALREADY INFERRED", "WILL BE APPLIED". The main drawback of this solution is that it would make the semantics of the network more complicated and the definition of the conditional probability distributions much more laborious. In POLA, we have adopted what appears to be a simpler and better solution.

POLA builds the Bayesian network incrementally from the AND/OR graph as the student enters actions, but keeps the two structures separately. For each new action that the student types, POLA determines which rules and which givens have been used to derive it. If there is more than one way to derive the student's action the analysis returns the set of possible derivations, each of which is a distinct but possibly overlapping set of rule applications and givens. Derivations are represented explicitly in the Bayesian network by what we call *derivation nodes*. After each new equation entered by the student the Bayesian network is extended with an *action node* that represents the equation and with a derivation node for each derivation producing the equation. The derivation nodes are linked to the action node through a leaky-XOR link matrix. Each derivation node is linked to the corresponding set of application nodes in the AND/OR graph through an AND link matrix. Each application node is linked to the corresponding rule node through a leaky-AND link matrix. All the priors in the network are set to the default value of 0.5. Figure 3 shows the structure of the network after the student has performed only the action *action1*, which can be derived in two different ways, represented by the derivation nodes *der1* and *der2*. The first derivation consists of the application *appl1* of *rule1* and the application *appl2* of *rule2*, while the second derivation consists of the application *appl3* of *rule3*.

This new topology still refers to the AND/OR representation of solution sets and paths, but uniquely defines the meaning of TRUE values for application, derivation and action nodes as "HAS BEEN PERFORMED". Moreover, the propagation time is reduced since the network is built incrementally. This is an important gain since POLA must perform the propagation of evidence on-line every time the student enters an action.

Although this new topology fixes the propagation problem presented by the OLAE Bayesian network, it still does not generate the correct probabilities for model tracing. The next section describes three problems that we encountered with this structure and how we solved them by introducing in the Bayesian network a new kind of node to explicitly represent the different solution sets available for a problem.

Representing solution sets probabilistically

The first problem is that this structure does not give enough relevance to the evidence provided by the student's actions in computing the probability of alternative derivations. Figure 3 shows how evidence is propagate backward through a first level of XOR connections (at the bottom) followed by a level of AND connections. Evidence is distributed among the nodes linked by the XOR relation in a way that is inversely proportional to the number of their AND parents. In our domain this means that if the student types an action that can be derived in more than one way, like *action1* in Figure 3, the most likely derivation is the one that requires the least number of steps (least complex), which in Figure 3 is derivation *der2*. This is a perfectly plausible rationale in absence of further evidence. On the other hand, if the student's next action provides evidence for a rule application that is related to one of the previous derivations, we want this derivation and the related application nodes to become the most probable, since they belong to the same solution set of the last rule application performed. This does not always happen with this topology. In Figure 4, for example, the next performed action *action2* provides indirect support for derivation *der1* through application *appl1* but, despite of this, *der2* still has higher probability then *der1*.

The second problem is that as soon as a rule node in the Bayesian network reaches an high probability, this is propagated to all the application nodes that are descendants of the rule node, including application nodes that belong to improbable derivations. Figure 5 shows an example of this inconsistent behavior. Let's imagine that the nodes in the figure are part of a larger network in which derivation node *der2* has high probability. This high probability propagates backward to the application node *appl3* and to rule node *rule2*. From here probability propagates forward to the application node *appl2*, which reaches high probability although it belongs to the improbable derivation *der1* and although there is no explicit evidence that the student has performed the corresponding rule application. This prob-

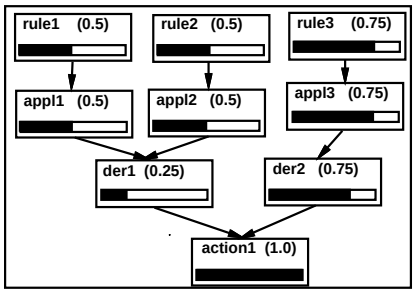

Figure 3: how the complexity of a derivation influences its probability

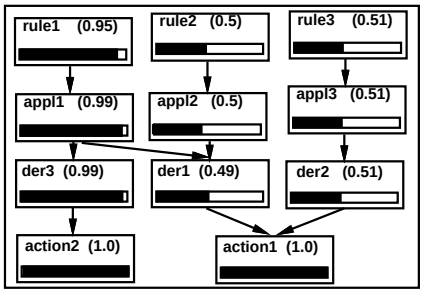

Figure 4: example of insufficient relevance of actions in determining the most probable derivation

lem arises because this topology, as the one used by OLAE, does not represent the fact that knowing a rule is different from being willing to apply it.

The third problem is that the Noisy-XOR link matrix between action and derivation nodes is semantically inappropriate.[2] We used it to encode our belief that very seldom students re-derive the same result twice and therefore given an action only one of the possible derivations should be TRUE. The problem is that the Noisy-XOR also encodes the inverse implication that, when more then one derivation for an action is true, the action is most probably false. This implication never actually occurs in our network, since by construction action nodes are always clamped to TRUE and the case of an action and all its derivations being TRUE at the same time is covered by the noise of the XOR relation, but this does not justify the fact that the Noisy-XOR does not represent the correct semantics in the network.

Solution nodes are the solution

We solved the three problems described above by introducing in the network *solution nodes* that explicitly represent the different solution sets for a certain problem. When a problem is selected by the student, POLA computes from the corresponding AND/OR graph the solution sets. In the AND/OR graph each node is marked with the solution sets in which it is used. In the graph in Figure 2, for instance, nodes R2, A4, F4, A6, F6, and A8 are marked as belonging to solution $SummMass$, nodes R3, A5, F5 and A7

[2] We thank Anthony Jameson for pointing out this problem.

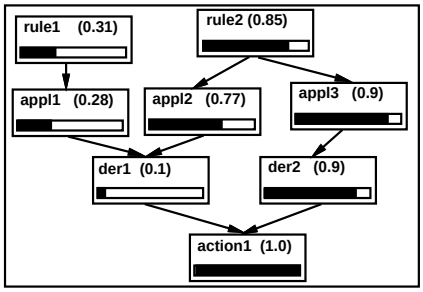

Figure 5: example of wrong propagation between rule and application nodes

as belonging to solution $SummWeight$ while all the other nodes are marked as belonging to both solutions.

Once the solution sets have been identified, POLA starts creating the Bayesian network adding a solution node for each solution set. The TRUE/FALSE values of a solution node indicate the probability that the student is pursuing the corresponding solution set. The solution nodes are then linked to a common ancestor, the *redundancy* node, whose values explicitly represent the probability that the student is following only one solution or more then one. The values of the redundancy node for a problem with two solutions are shown in Figure 6. The priors of the redundancy node allow one to express the presumably low probability that results are derived more than once, since the fact that the student is pursuing multiple solutions is directly related to the fact that the student performs two derivations of the same result. For each new student's action an action node and the corresponding derivation nodes are added to the network. An additional derivation node, like *other1* and *other2* in Figure 6, is inserted for each action, to encode the probability that the student has performed the action by guessing or copying from other solutions. Since the probability of mutually exclusivity of derivations is encoded into the *redundancy* node, the link matrix between an action node and its parent derivation nodes can now be defined by an OR link, eliminating the inappropriate implications entailed by the Noisy-XOR link.

At this point POLA retrieves in the AND/OR graph the solution sets to which each derivation belongs. Each application node in a derivation is connected to the corresponding solution node and to the node corresponding to the applied rule, as shown in Figure 6 and in Figure 7. The conditional probability distribution for an application node is defined by a leaky-AND matrix, that represents two assumptions. First, that it is possible that the rule has not been applied even though the rule and the solution are TRUE. Second, it is possible that the application has been performed even though either the solution or the rule are FALSE, by guessing or copying from other solutions. In interpreting the meaning of the Noisy-AND link between application, rule and solution nodes, it is important to keep in mind that application nodes are inserted in the network only when a student's action provides evidence for them. Therefore the

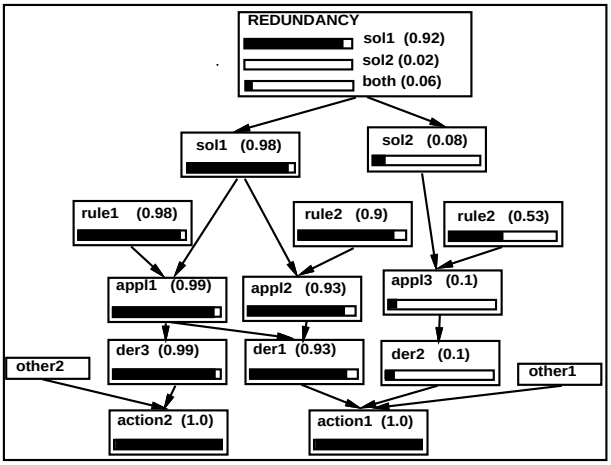

Figure 6: solution nodes augment the relevance of the student actions

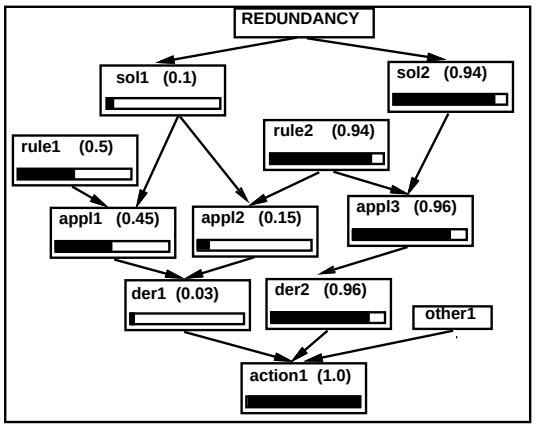

Figure 7: solution nodes regulate the propagation of evidence between rule and application nodes

AND relation encodes the fact that, when an action that may require a certain rule application has been typed, if the rule is known and the solution set to which the application belongs is probable, then the rule has probably been applied. The other possible meaning of the AND relation, namely that if a rule and a solution are highly probable then the application of the rule in the context of that solution will probably happen, is not represented by our network. As a matter of fact the decision of building the network incrementally has been taken to avoid modeling the distinction between inferences that have been made in the past and inferences that will be made in the future. A sound representation of this distinction will require an explicit representation of time in the network.

The links between a solution node and the application nodes belonging to the corresponding solution set provide an explicit representation of the student's intention to apply a rule within a specific solution set. These links emphasize the relevance of the evidence coming from student's actions, thus solving the first problem mentioned in the previous section. Figure 6 shows how the addition of solution nodes $sol1$ and $sol2$ makes derivation $der1$, for which indirect evidence is provided by $action2$, more probable then $der2$, adjusting the proportion of the probabilities of $der1$ and $der2$ shown in Figure 4. A direct assessment of the probability of each solution is immediately available from the values of $sol1$ and $sol2$.

Moreover, the introduction of solution nodes solves also the problem of the wrong propagation of evidence from rule nodes to application nodes. In fact, the links between application and solution nodes prevent the high probability of a rule node from spreading to the application nodes that belong to unlikely solutions, modeling the fact that a rule application node can have high probability only if it is part of an highly probable solution or if there is direct evidence that the rule has been applied. Figure 7 shows the same situation represented in Figure 5 encoded in the new topology. In Figure 7 the low probability of node $sol1$ prevents the high probability of node $rule2$ to propagate to node $appl2$ through the leaky-AND link matrix, fixing the too high probability reached by $appl2$ in the network of Figure 5.

Current version of POLA

The addition of solution nodes to the Bayesian network used by POLA allows the system to generate reasonable probabilistic predictions about the solution that the student is following even when multiple solutions are consistent with the student's actions. Moreover, the network still propagates to rule nodes the evidence coming from student's actions, maintaining for POLA the capability that OLAE had for performing knowledge tracing.

As an illustration of how POLA works, we now summarize the steps that POLA performs to build the network in Figure 8. The network is built after the student has entered as her first action the equation $Wa = 775$, in the attempt to solve the problem in Figure 1.

Before the student starts solving the problem, the only nodes in the Bayesian network are the root node *redundancy* and its children, solution nodes $s1$ and $s2$, representing solution $SummMass$ and solution $SummWeight$ respectively. As soon as equation $Wa = 775$ is entered, POLA retrieves from the AND/OR graph the two different sets of application nodes that can be used to derive the equation, namely nodes $\{A1, A2, A3, A4, A6, A8\}$ and $\{A1, A2, A3, A5, A7\}$ in Figure 2. Then it adds to the network an action node, labeled $Wa = 775$ in Figure 8, that represents the entered equation, a derivation node for each set of applications (nodes $der1$ and $der2$ in Figure 8) and an additional derivation node (*other1* in the figure), whose prior represents the probability that the action has been derived in an unorthodox way.

The derivation nodes are linked to the action node through an OR link. Each derivation node is linked through an AND link matrix to the application nodes in the corresponding set. Application nodes are the nodes at the third level in Figure 8. Each application node is linked through a leaky-AND link

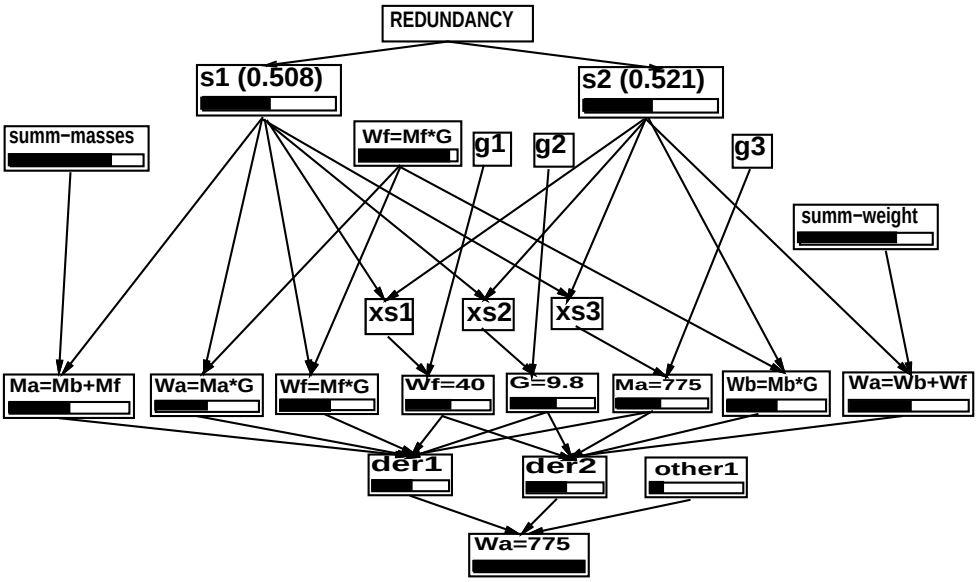

Figure 8: state of the Bayesian network after the student's input "Wa=775"

matrix to the corresponding rule node and to the solution node representing the solution set to which the application belongs. Rule nodes in Figure 8 are the nodes labeled *summ-masses*, *summ-weight*, *W=M*G*, $g1$, $g2$ and $g3$.

When an application belongs to two or more solutions, a node (like $xs1$, $xs2$ and $xs3$ in Figure 8) is interposed between the application node and the solution nodes and its conditional probability distribution is defined by an OR link matrix. All the prior probabilities in the network are set to the default value of 0.5.

The action node corresponding to $Wa = 775$ is then clamped to 1.0 and propagation is run. Although it is implausible that $Wa = 775$ be the first action that the student performs, we are using this situation to show how the Bayesian network assigns probabilities to alternative solution sets when only ambiguous actions are available. Although the action $Wa = 775$ belongs to both solution $SummMass$ and solution $SummWeight$, there is a slight difference in probability between the corresponding solution nodes $s1$ and $s2$. The difference is due to the fact that derivation *der1*, that is part of solution $s1$, has smaller probability than *der2* because it consists of fewer rule applications. This behavior of the network favors solutions that require less steps, in absence of further evidence. The behavior can be changed by changing the priors of the values of node *redundancy* to favor the solution that is a priory believed to be more probable.

Figure 9 shows an example of how probabilities change in favor of the solution $SummMass$, represented by node $s1$, when explicit evidence to support this solution is provided by the action $Mf = 40/9.8$. As far as the assessment of rule knowledge is concerned, the network correctly assigns high probability to the rule node *summ-masses* connected with $s1$, while increases only slightly the probability of *summ-weight*, connected to the less probable solution node $s2$. Rule nodes *W=M*G*, $g1$, $g2$ and $g3$ have very high probability since they belong to both solutions. Finally, Figure 9 shows how probabilities are coherently assigned to the application nodes attached to the rule node *W=M*G*. The two application nodes *Wa=Ma*G* and *Wf = Mf*G* correctly have probability much higher than *Wb=Mb*G* since they are associated with the most probable solution $s1$.

Conclusions and future work

The previous sections described how we have modified OLAE, an off-line assessment system that performs probabilistic knowledge tracing, to develop POLA, a probabilistic student modeling framework that performs both knowledge and model tracing.

POLA generates predictions about the student's current line of reasoning without using heuristics, even when the solution space is large. An AND/OR graph provides a compact representation of all the available solution sets and solution paths and allows POLA to keep track of the student's progresses in the solution space. A Bayesian network built incrementally from the AND/OR graph and from the student's actions handles the uncertainty created by actions that are consistent with multiple solution paths.

POLA represents a new approach to applying Bayesian networks to student modeling, since the other existing efforts to use probabilistic reasoning in student modeling have been focused on knowledge tracing only (op. cit.).

We have started to evaluate the accuracy of the model tracing generated by POLA using protocols collected through OLAE from 3 students solving the example problem presented in this paper and from 2 students solving a more complex problem involving 4 different solution sets and 18 primitive equations. The predictions generated by

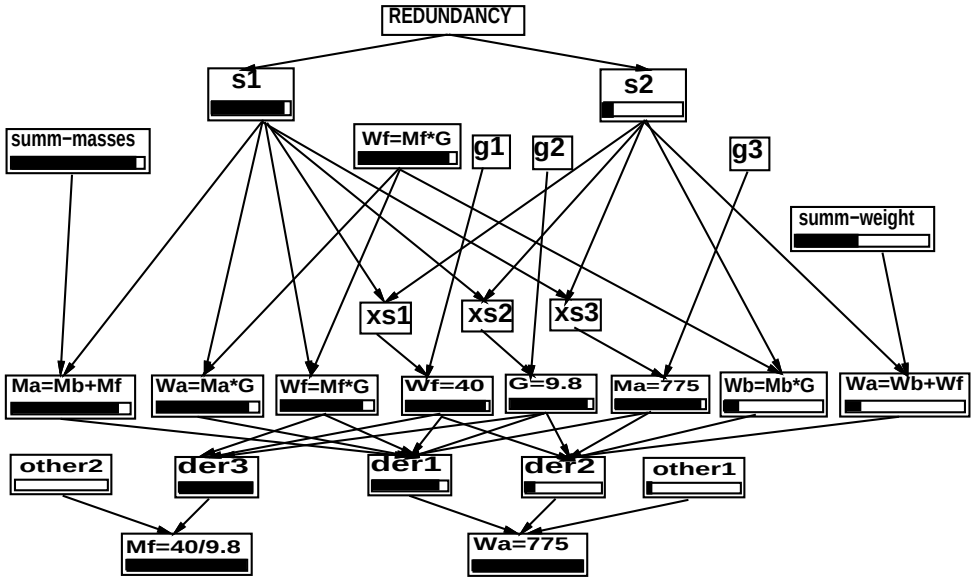

Figure 9: state of the Bayesian network after the student's input "Mf=40/9.8"

POLA on these 5 protocols have always been consistent with the solution that each student actually followed, but we plan to perform a more formal evaluation with more protocols and more complex problems.

Another step in the future of POLA will be to define more precise prior probabilities for rule, solution nodes and for the *redundancy node*, through interviews with physics experts and the observation of the students behavior during the interaction with POLA. Later, when large samples of students are available, these subjective estimates can be replaced by empirical ones.

References

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

Corbett, A., and Anderson, J. 1992. Lisp intelligent tutoring system: Research in skill acquisition. In Larkin, J., and Chabay, R., eds., *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared goals and complementary approaches*. Hillsdale, NJ: Erlbaum. 201–238.

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., eds., *Computers as Cognitive Tools*. Hillsdale, NJ: Lawrence Erlbaum.

Duncan, D.; Brna, P.; and Morss, L. 1994. A bayesian approach to diagnosing problems with prolog control flow. In *Proceedings of the Fourth International Conference on User modeling*.

Gitomer, D.; Steinberg, H.; S., L.; and Mislevy, R. J. 1995. Diagnostic assessment of troubleshooting skill in an intelligent tutoring system. In Nichols, P.; Chipman, S.; and Brennan, R., L., eds., *Cognitively diagnostic assessment*. Hillsdale, NJ: LEA.

Martin, J., and VanLehn, K. 1995. A bayesian approach to cognitive assessment. In Nichols, P.; Chipman, S.; and Brennan, R. L., eds., *Cognitively diagnostic assessment*. Hillsdale, NJ: LEA.

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

Mislevy, R. J. 1995. Probability-based inference in cognitive diagnosis. In Nichols, P.; Chipman, S.; and Brennan, R., L., eds., *Cognitively diagnostic assessment*. Hillsdale, NJ: LEA.

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

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*, 426–432.

Sime, J. 1993. Modelling a learner's multiple models with bayesian belief nets. In *Proceedings of the 1993 World Conference on AI and Education*.

Singley, M. K. 1990. The reification of goal structures in a calculus tutor: Effect on problem-solving performance. *Intelligent Learning Environments* 1(2):102–123.

Villano, M. 1992. Probabilistic student models: Bayesian belief networks and knowledge space theory. In *Proceedings of the 2nd International Conference on Intelligent Tutoring Systems*. Berlin: Springer-Verlag.