A student modeling technique for problem solving in domains with large solution spaces

Cristina Conati
Intelligent Systems Program
Learning Research and Development Center
3939 O'Hara St.
University of Pittsburgh
Pittsburgh, PA 15260
tel. 412-624-7536
conati@pogo.isp.pitt.edu

Kurt VanLehn
Computer Science Department
Learning Research and Development Center
3939 O'Hara St.
University of Pittsburgh
Pittsburgh, PA 15260
tel. 412-624-7458
vanlehn@cs.pitt.edu

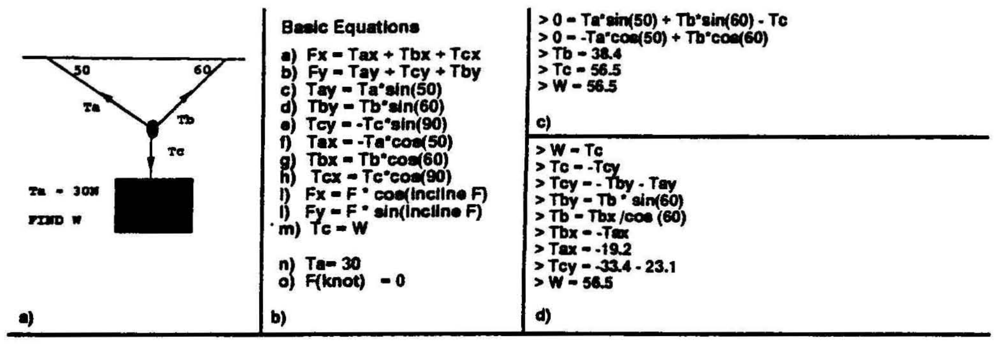
Poster Proposal

Areas of interest: student modeling, cognitive diagnosis

1 Introduction

In order for a coached practice environment to give effective advice, it must at minimum determine whether a student's actions lead eventually to a correct solution. Some model tracing tutors, for instance, give immediate feedback whenever the student's action does not lie along a known solution path. Even if the coach does not give immediate feedback, it needs to determine what path the student is on before it decides what it should do in response to the student's action[1].

However, in some task domains, there are too many correct solution paths to represent them explicitly. For instance, when solving the college physics problem shown in Figure 1a, there are 11 primitive equations that can be applied (Figure 1b). Figures 1c and 1d show two of the more than 11![1] correct derivations. When preferred problem solving strategies can be identified, an approach to handle this complexity is to limit the acceptable solution paths to those generated by the preferred strategies (e.g., [2, 4]). Another approach is to provide an environment that supports problem solving through a carefully designed interface, without providing tutorial guidance (e.g.,[5]). We are developing a coached practice environment for a certain class of domains whose problems have many distinct acceptable solution paths, and it would be too constraining to prune to a predefined subset. This poster presents our current research on the student model component and hint selection strategies for such an environment.



```
Basic Equations
a) Fx = Tax + Tbx + Tcx
b) Fy = Tay + Tcy + Tby
c) Tay = Ta*sin(50)
d) Tby = Tb*sin(60)
e) Tcy = -Tc*sin(90)
f) Tax = -Ta*cos(50)
g) Tbx = Tb*cos(60)
h) Tcx = Tc*cos(90)
l) Fx = F * cos(incline F)
l) Fy = F * sin(incline F)
m) Tc = W

n) Ta= 30
o) F(knot) = 0
```

```
> 0 = Ta*sin(50) + Tb*sin(60) - Tc
> 0 = -Ta*cos(50) + Tb*cos(60)
> Tb = 38.4
> Tc = 56.5
> W = 56.5
```

```
> W = Tc
> Tc = -Tcy
> Tcy = - Tby - Tay
> Tby = Tb * sin(60)
> Tb = Tbx /cos (60)
> Tbx = -Tax
> Tax = -19.2
> Tcy = -33.4 - 23.1
> W = 56.5
```

Figure 1:

2 The AND/OR representation of multiple solution paths

The task domain we are currently working in is classical physics. In physics, a correct solution consists of a derivation of a sought quantity via applications of generic equations (e.g., Newton's law) and substitution of given quantities. Such a derivation can be represented by an AND/OR graph (Figure 2) where the AND nodes (circles) are equation applications and the OR nodes (diamonds) are derived quantities. Every correct solution path corresponds to a traversal of this AND/OR graph. For instance, the path of Figure 1c corresponds to applying first the equations corresponding to nodes A1, G2, A3, A8, A9 and A10, then the equation corresponding to nodes A2, G2, A4, A5, A6 and A7, then to derive the quantities corresponding to nodes O7 and O10 and finally to apply the equation corresponding to A11 to derive the sought quantity represented by O11. In many other task domains (e.g., theorem proving, algebraic equation solving, thermodynamics), the solution is also a derivation that can be represented by an AND/OR graph, and a solution path

[1]This estimate assumes the student writes only primitive equations. When combinations of primitive equations and givens are written, as in Figures 1c and 1d, the number of correct solution paths is much higher.

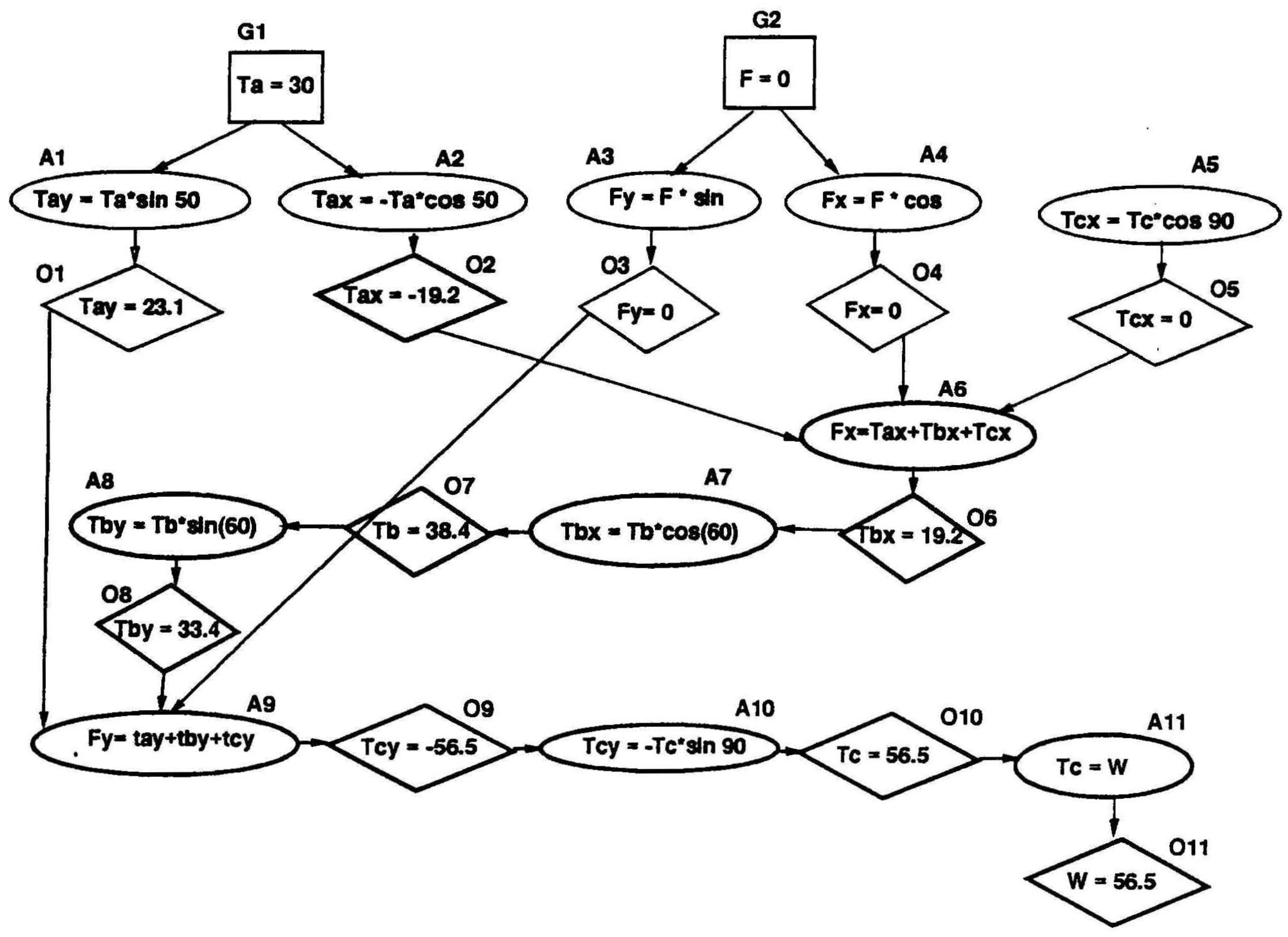


Figure 2:

is a traversal of this graph. Each AND node corresponds to the application of a domain inference rule, and each OR node corresponds to a proposition.

Our system constructs an AND/OR graph for each problem in advance, from a knowledge base of production rules that encodes the physics knowledge necessary to solve problems in Newtonian physics. As the student enters actions, the system checks off the AND nodes that correspond to the inferences that the student has made and the OR nodes that correspond to the quantities already derived. If the student makes an inference that is not part of the domain knowledge base, there will be no corresponding set of AND nodes for the student's action and the system can notify the student of the mistake.

Since there are so many correct solution paths available, when the student makes a mistake or asks for help, the system cannot immediately tell what the appropriate next step for the student should be. In fact, even if the tutor keeps track of the path the student has followed so far, many possible correct continuations are available. Our hypothesis is that an appropriate next step to suggest would be one of the inferences (AND nodes) that is adjacent to the part of the graph that has been checked off so far. If more then one of these inferences are available, which one to select will be determined by heuristics and Bayesian reasoning.

3 Hint selection

The tutorial module is integrated with OLAE [3], an assessment tool that collects data from students solving problems in introductory college physics. OLAE provides an interface that presents problems and allows the student to solve them by drawing force diagrams and entering algebraic equations. We have developed an algorithm that generates all the algebraically correct equations the student can enter, associates them with nodes in the AND/OR graph and uses this structure to verify the validity of the student input and to select hints when the student asks for help.

Given the problem of Figure 1a, let's suppose that the student types[2]

```
0 =  Ta*sin(50) + Tb*sin(60) + Tc.
```

The system extracts all the inferences and intermediate results that have been implicitly performed to write this equation, namely those corresponding to nodes A10, A9, A8, A1, A3, G2, and 03 in Figure 2.

Let's suppose that the student asks for help now. The system collects all the AND nodes adjacent to the inferences the student has already applied: G1, A11 and A7 (nodes corresponding to givens are considered special instances of AND nodes). This set of nodes is called the *hinting set* and the system uses heuristics to select a node from it.

G1 represents a given and the tutor assumes that the student knows it. It would make sense to give A11 as a hint if there was evidence that the student was following a top-down approach, trying to write the primitive equations closely connected to the sought quantity. There is no such evidence, since the student has written only one equation and has already substituted values in it. The system therefore applies the default heuristic of giving a hint about an equation that allows the student to derive one of the variables still present in the student's equation, namely the equation represented by node A7.

In the real protocol, instead of asking for help, our student types

```
0 =  Ta*sin(50) + Tb*sin(60).
```

This equation does not correspond to any node in the AND/OR graph and the tutor detects an error. From the analysis of the variables present in the equation, the tutor guess that the student is trying to compute Tb. The fact that node A7, which allows the student to derive Tb, is part of the hinting set adds evidence to the tutor's guess. The tutor can therefore stop the student and give hints about the correct inference to use to find Tb, namely applying $Fx = Tax + Tbx + Tcx$.

As the example shows, the topology of the AND/OR graph and the student's actions are used by the tutor to formulate guesses about the student line of reasoning and to elaborate hints when the student needs support. In the future, we may add the capability to ask the student about the validity of the tutor's guesses.

4 Future work

Our first goal is to integrate the heuristic approach for hint selection with the use of probabilities. Olae already has the functionality to interpret the AND/OR graph as a Bayesian Network and to propagate probabilities from actions to rule nodes following the links in the solution graphs. We are planning to use these probabilities in the hint selection process. Moreover, probabilities could be used to exploit structural links existing among rules in the knowledge base, indicating how the

[2]The equations used in this example are taken from the protocol of a student using Olae.

knowledge of a certain rule can affect the knowledge of another. For example, it is quite probable that a student who knows the rule for projecting a vector onto the x axis will know the equivalent rule for the y axis.

We are planning to use real students to evaluate the hint selection heuristics developed, in order to have empirical data to refine them and to devise new ones.

References

[1] Anderson, J.R., Corbett, A.T., Koedinger, K., Pelletier, R. *Cognitive Tutors: Lessons learned,* Cognitive Science, in press.

[2] Corbett, A.T., Anderson, J.R. *LISP Intelligent Tutoring System: Research in Skill Acquisition,* (1992). In Larkin, J.H. & Chabay, R.W. (Eds), Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared goals and complementary approaches, pp. 201-238. Hillsdale, NJ: Erlbaum.

[3] Martin, J., VanLehn, K. *OLAE: Progress toward a multi-activity, Bayesian student modeler,* (1993). In Brna, P., Ohlsson, S. & Pain, H. (Eds), Proceedings of the 1993 World Conference on AI and Education, pp. 426-432. Charlottesville, NC: Association for the Advancement of Computing in Education.

[4] Singley, M. K. *The Reification of Goal Structures in a Calculus Tutor: Effect on Problem-Solving Performance,* (1990). Intelligent Learning Environments, vol 1(2), pp 102-123.

[5] Foss, C. L. *Productive thrashing in a computerized tutoring system,* (1986). Proceedings of the Third International Conference of Artificial Intelligence and Education.