

Towards an Intelligent Tutoring System for Propositional Proof Construction

Marvin CROY ^{a,1}, Tiffany BARNES ^b and John STAMPER ^b

^a *Department of Philosophy, The University of North Carolina at Charlotte*

^b *Department of Computer Science*

Abstract. This article reports on recent efforts to develop an intelligent tutoring system for proof construction in propositional logic. The report centers on data derived from an undergraduate, general education course in Deductive Logic taught at the University of North Carolina at Charlotte. Within this curriculum, students use instructional java applets to practice state-transition problem solving, truth functional analysis, proof construction, and other aspects of propositional logic. Two project goals are addressed here: 1) identifying at-risk students at an early stage in the semester, and 2) generating a visual representation of student proof efforts as a step toward understanding those efforts. Also discussed is the prospect for developing a Markov Decision Process approach to providing students with individualized help.

Keywords. Proof construction, intelligent tutoring system, Markov decision processes, visualization.

Introduction

Intelligent tutoring systems generally involve three component models: one of the student, one of the teacher/pedagogical technique, and one of the subject matter [1]. Each model presents its own challenge in respect to construction and effective use. One issue in respect to student models concerns their content, i.e., what student characteristics should be included, what aspects of performance should be represented, etc. One challenge in respect to pedagogical technique concerns the form and timing of help [2]. Perhaps the most challenging task concerns the explication of expert knowledge concerning the subject matter and how this knowledge relates to pedagogical technique. The number of hours required to achieve this task are normally prohibitive and certainly constitute one of the most costly components of an intelligent tutoring system [3].

¹ Corresponding Author: Marvin Croy, Department of Philosophy, University of North Carolina at Charlotte, Charlotte, NC, 28223, USA.

The activities reported here address a number of these challenges in respect to teaching deductive proof construction in propositional logic. In respect to student models, our aim is to discover variables that will identify the most at-risk students at an early stage in the course. Once we can reliably predict which students might need special help, the challenge is to design activities that will provide this individualized assistance. One objective is to supply individualized help as students use instructional programs that facilitate the learning of proof construction. The general nature of this assistance is guided by visual representations of student problem solving efforts, and by a novel approach to leveraging past student data to provide intelligent assistance. This approach is based on using Markov Decision Processes (MDPs) to represent student paths connecting premises and conclusion [4].

1. The Course Curriculum

Our Deductive Logic serves general education students during their first two years of undergraduate study. This course begins with state transition problem solving. Students practice a number of classic problems such as the Towers of Hanoi, Water Jugs, and Jealous Husbands [5]. In this context a problem is defined in terms of a starting state, goal state, and transition rules. This readily maps on to deductive proof problems via a premise set (starting state), a conclusion (goal state), and a set of valid forms of inference/replacement (transition) rules [6]. In particular, working backwards from goal to starting state translates into working from conclusion to premises, and this provides an alternative strategy for discovering proofs [7,8]. The mastery of proof construction and valid inference-making is central to the course. The skills developed in proof construction are made use of in subsequent course topics, such as SQL-type database searching, decision making in the context of structured documents (e.g., executing Internal Revenue Service tax form instructions), and argument analysis in natural language. This point is crucial since failure to master deductive inference and proof construction becomes evident when students reconstruct natural language passages via deductive patterns of inference.

Students face examples similar to the example shown in Table 1. Students are expected to recognize premise/conclusion relationships, reformulate statements, explicate implicit premises, identify assumed synonyms, delete unnecessary information, and fit the argument to a valid form where possible (which sometimes requires the application of rules of replacement). In particular, when students discover proofs by working backwards from conclusion to premises, they postulate sub-goal expressions that, if ultimately justified, lead directly to the proof's conclusion. Mastering this process facilitates the ability to identify implicit premises in natural language arguments. In the example given in Table 1, notice that the second premise ("If x can be attained only through the methods of the natural sciences, then x is based on generally observable facts") is not given in the original argument and must be explicated by the student. Familiarity with the rule of Hypothetical Syllogism aids in making this explication. While the argument given for reconstruction may seem trivial, this is exactly the kind of problem that weaker students fail to complete.

Table 1. Sample argument and components of its reconstruction.

Argument and Reconstruction	Basis for Reconstruction
<p>“Finally there is the triumphant idea of positivism, that valid knowledge can be attained only through the methods of the natural sciences and hence that no knowledge is genuine unless it is based on generally observable facts” [9].</p>	<p>1) Distinguish premise(s)/conclusion (conclusion indicator = ‘hence’)</p> <p>2) Fill in information (filled in referent for ‘it’) (supplied implicit premise)</p> <p>3) Delete information (Deleted ‘Finally there is the triumphant idea of positivism, that’ on basis of being irrelevant to the inference pattern)</p> <p>4) Reform statements into conditionals (universal subjects / necessary conditions)</p> <p>5) Assumed Synonyms (‘valid knowledge’ = ‘genuine knowledge’)</p>
<p>Given premise: If A then B Implicit premise: If B then C Conclusion: If A then C</p>	<p>A = ‘x is valid knowledge’ B = ‘x can be attained only through the methods of the natural sciences’ C = ‘x is based on generally observable facts’</p>

Throughout the course students make use of instructional Java applets². These programs not only provide opportunities for students to develop and hone relevant skills, but they also support the timely collection of data on student performance. This is particularly important during the early weeks of the semester. During this period, students use applets to complete state transition problems, carry out truth functional evaluation, and practice the application of deductive proof rules. Two applets are central to this curriculum. “Justified Thought” (JT) provides practice with the inference/replacement rule set. JT uses mal-rules and complex instantiation to build a deep understanding of rule applications. The program provides three levels of increasing difficulty for both inference (implicational) and replacement (equivalence) rules. Students must judge whether particular expressions do or do not conform to various rule patterns. In level one, some rule pattern always fits the given expression, which JT constructs by instantiating rule forms with simple statement letters. In higher levels of JT, some expressions fit none of the rule patterns shown. In these cases, the expression fits a mal-rule, a deviant of the actual rule. Moreover, higher levels of JT use complex expressions to instantiate rule variables.

Practice with JT occurs in parallel with work in “Deep Thought” (DT), a Java applet that provides a graphic environment for building proofs. DT supports both working forwards and backwards and, along with JT, maintains detailed records of student efforts. Developing prowess in constructing proofs provides the main challenge of the course. When investigating the difficulties students have in learning proof construction, attention is naturally drawn to two main skills: rule application and

² Students access these applets via a course management system such as BlackBoard that supports applet use by means of a number of additional resources. Direct, unsupported access to the applets is available via <http://itsxserve.uncc.edu/philosophy/talle/applets/applets.html>.

strategic planning. Logic textbooks almost invariably isolate these activities by presenting rule application exercises prior to full proof problems. Nevertheless, it should be understood that the selection and ordering of rule applications can be shaped by strategic considerations. One question of interest is how rule application and strategy selection interact, whether positively or negatively. Obviously, there is a positive interaction when strategy selection (e.g., breaking expressions down into components) suggests the application of various rules (e.g. Simplification). However, when strategic thinking proposes a highly useful intermediate expression that could lead to the conclusion, similarities between this proposed sub-goal expression and some premise (or previously derived expression) may produce attempted misapplications of rules. For instance, when 'not A' constitutes a goal expression, students are tempted to misapply the rule of Simplification to the premise expression 'not (A and B)'. Here, the main connective of the premise expression is actually a negation and not a conjunction as required for application of Simplification.

Another aspect of our efforts concerns the question of what contributes to proof problem difficulty and how these conditions relate to the connection between proof construction and argument reconstruction. As suggested in the sample argument in Table 1, argument reconstruction often involves the removal of irrelevant information. The analog within proof construction is superfluous premises. It is unfortunate that logic textbooks do not routinely present proof problems with unnecessary premises. Particularly when the superfluous premises contain components similar to the components of other premises or the conclusion, problem difficulty increases, and the required distinction between the relevant and the irrelevant provides part of the bridge to argument reconstruction. Our visualization method helps identify other problem areas in proof construction that we can later analyze for their impact on argument reconstruction.

2. The Search for Early Predictors

The first data set analyzed includes records from 85 students enrolled in three sections of Deductive Logic during 2007. Data from these students include: 1) scores on a pre- and post-test (number correct out of 25 multiple-choice items, primarily focused on validity judgments and deductive inferential tasks), 2) performance measures from a state-transition problem (Jealous Husbands: number of incorrect moves), 3) success rate for level 3 performance on truth functional evaluation of complicated logical expressions, 4) success rate for levels 1, 2, and 3 of the JT rule practice applet, 5) two exam scores (mid-term and final), and 6) course grade (expressed as percent of total possible points on exams plus additional assignments).

In order to correctly identify at-risk students early in the semester, we searched for items from our data set that could provide a significant correlation with exam scores and final course grade. Pearson correlations were run using SPSS statistical software to analyze the correlation between the exam scores, final course grade, and the following predictors: pre-test score, Jealous Husbands Puzzle success (measured by number of solution steps and incorrect moves), Truth Functional Analysis success rate, and Justified Thought success rates for levels 1, 2, and 3. The resulting correlations can be seen in Table 2.

Table 2. Correlations of potential predictors with course performance.

<i>Course Performance</i>	<i>Pre-test</i>	<i>Jealous Husbands Puzzle</i>	<i>Truth Functional Analysis</i>	<i>JT Rule Exercise Level 1</i>	<i>JT Rule Exercise Level 2</i>	<i>JT Rule Exercise Level 3</i>
Exam 1	.359 (*)	.072	.238 (*)	.440 (**)	.484 (**)	.617 (**)
Exam 2	.129	-.413 (**)	.198	.525 (**)	.447 (**)	.302 (*)
Exam Total	.271	-.196	.249 (*)	.566 (**)	.533 (**)	.541 (**)
Course Grade	.242	-.305	.173	.573 (**)	.559 (**)	.491 (**)

* significant at .05 level

** significant at .01 level

From the results, there are two interesting points that immediately stand out. First, the pre-test is not the most reliable predictor for success in the class. Although pre-test performance had some correlation with exam 1 (significant at $p=.05$, but not at $p=.01$), it did not correlate well with Exam 2 or final course grade. This result was surprising, since we originally felt the pre-test would predict class performance. However, the lower correlation between the pre-test and the final course grade is a healthy indicator that learning can occur and student performance does not depend entirely on prior preparation.

The other main finding from these statistics is how well the JT success rates correlate with exam scores and course grade. Based on this, we created metrics using the JT success rates as our early predictors. The goals of our metrics were to identify as many at-risk students as possible while minimizing the number of students incorrectly assigned to this category. To further this investigation, we selected a past class (Spring 2007, with 30 students) to serve as a starting case (training set), and the most recent class (Summer 2007, with 20 students) as a test case (trial set). In the training set, low course performance was defined in terms of a course grade of 65% or lower. We determined cutoff values for each of the three levels on JT (JT1, JT2, and JT3), and examined how well they identified low-performing students.

The extent to which various success rates on JT correctly identified students with low performance is shown in Table 3. This table shows the number of students who were correctly classified as low-performing (course grade of less than sixty-five percent) according to five different metrics based on JT success rates: less than 90% on JT1, less than 70% on JT2, less than 75% on JT3, a combination of each of these three, and a combination of only JT2 with JT3 performance. Table 4 also shows the number of students misclassified by each of these metrics. Misclassification occurs as false negatives (students who scored above the metric success rate but who were actually low-performing) and as false positives (students who scored below the metric success rate but who were not low performing). The results show that the best results occur for a combination of JT2 with JT3. This metric correctly classifies 25 students, generates no false negatives, and identifies only 5 false positives. False positives, students inaccurately identified as being at-risk, are not particularly troublesome, since these students would likely be given more assistance than actually required. False negatives, at-risk students treated as not being so, are of more concern, and their low numbers here are encouraging.

Table 3. Number of low-performing students identified by JT performance levels (Training Set).

	<i>JT1 Success Rate < 90%</i>	<i>JT2 Success Rate < 70%</i>	<i>JT3 Success Rate < 75%</i>	<i>JT1, 2, & 3</i>	<i>JT2 & JT3</i>
Correctly Categorized	22	27	22	23	25
False Negatives	5	2	3	0	0
False Positives	3	1	5	7	5

The results of testing these metrics against data from the most recent logic course are shown in Table 4. Once again, many more correct than incorrect classifications are made using any of the previously defined metrics, and most misclassifications are false positives. However, the metric that combines performance on JT2 with JT3 was slightly outdone by the metric that focuses only on JT2. In sum, the metrics defined in the training set do a good job of classifying students in ways that, from a pedagogical standpoint, reliably identify at-risk students. The best of the JT metrics serve to predict students who are candidates for special attention or assistance. Answering questions about the nature of that assistance now becomes even more crucial.

Clearly, the first step in addressing at-risk students is to ensure that these students complete practice with JT, and particularly JT2, with at least 70% correct. However, the next step in assisting students is not as clear. In the next section, we discuss one alternative for providing proof hints within DT that will specifically benefit at-risk students.

3. Visualizing the Nature and Variation of Student Proofs

Visual maps of student proof efforts serve to shape new questions about the how and why of student thinking. In addition, graphic representation provides a snapshot of the range of variation in student efforts. A greater range of variation increases the challenge of providing individualized help.

We use Markov Decision Processes to visualize student proof attempts, as in [4]. A Markov decision process (MDP) is defined by its state set S , action set A , transition probabilities P , and a reward function R [10]. On executing action a in state s the

Table 4. Number of low-performing students identified by JT performance levels (Trial Set).

	<i>JT1 Success Rate < 90%</i>	<i>JT2 Success Rate < 70%</i>	<i>JT3 Success Rate < 75%</i>	<i>JT1, 2, & 3</i>	<i>JT2 & JT3</i>
Correctly Categorized	13	17	15	12	15
False Negatives	4	1	1	1	1
False Positives	3	1	5	7	4

probability of transitioning to state s' is denoted $P(s' | s, a)$ and the expected reward associated with that transition is denoted $R(s' | s, a)$. For a particular point in a student's proof, our method takes the current premises and the conclusion as the state, and the student's input as the action. Therefore, each proof attempt can be seen as a graph with a sequence of states (each describing the solution up to the current point), connected by actions. We combine all student solution graphs into a single graph, by taking the union of all states and actions, and mapping identical states to one another.


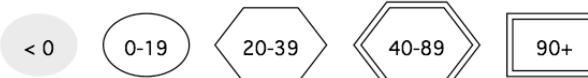
Once this graph is constructed, it represents all of the paths students have taken in working a proof. Typically, a reinforcement learning technique such as value iteration Bellman backup [10] is used to assign reward values to all states in the MDP. The rewards for each state indicate how close to the goal a state is, while probabilities of each transition reveal the frequency of taking a certain action in a certain state. Once the rewards are calculated, the "best" solution to the problem corresponds to taking a path through the graph, choosing nodes of maximum reward to reach the goal state [10]. For this paper, just one step of value iteration was performed to cascade goal values through the MDP.

This experiment uses data from the Spring 2005 and Fall 2005 semesters of the first author's Deductive Logic course, with a total of 69 students. After lectures on the topic and practice using JT, students use DT to solve 12 logic proofs as homework. We extracted 69 students' attempts at solutions to DT proof 1.3. Of these 69 attempts, 52 (75%) were successful proofs, while 17 were incomplete (25%). After cleaning the data, we load the proofs into a database and build a graph for the data. We then set a large reward for the goal state (100) and penalties for incorrect states (10) and a cost for taking each action (1). Setting a non-zero cost on actions penalizes longer solutions (but we set this at 1/10 the cost of taking an incorrect step). These values may need to be adjusted for different problems.

We then created an MDP as described above for the proof data, resulting in a set of 193 states and associated reward values, and 272 actions. Using Excel®, we assigned labels to each state in the MDP (using the latest premises added), colors for errors, state values, and action frequencies, and prepared the data for display. We used GraphViz to display the output. Table 5 shows the legend for nodes and edges. After graphing the MDP, we continually refined the visualization to explore questions about the data.

Figure 1 shows the MDP restricted to the actions that at least 4 students have taken (since the overall MDP is too large to view here). Actions are labeled with a B for backwards steps and F for forward steps, followed by a dash and the rule used. The starting state of the problem contains 3 given premises: $(\sim A \vee B) \supset (C \& D)$, $A \supset E$, $\sim E$, and the result: $?D$. In DT, the premises to be proven are denoted with $?$, and when a student works backwards, the original result is shown to be justified by the intermediate

Table 5. Legend for MDP edges and nodes

Edges (Values=Frequency)	Nodes (Values=Rewards)
	

expressions the student postulates, and the intermediate expressions become new provisional postulates. For example, many students work backward from ?D using simplification (B-SIMP) to postulate the intermediate expression ?C&D. As shown at the top of Figure 1, many students take this as a first step. In this step, the ? is removed from D and remains on ?C&D until it is justified.

In Figure 1, we see that there are a number of ways students solve problem 1.3. The leftmost part of Figure 1 shows the students who worked primarily backward in solving 1.3, while the central area shows students with forward and mixed approaches. The rightmost part shows an abandoned attempt to work backwards via MP. From this diagram, we note that students working primarily backward seem to have much more trouble determining that they must prove $\sim A$ than those students who started using a forward approach. We are planning to build a tool that enables instructors to interactively view student solutions and understand student thinking and approaches to problems.

Figure 2 shows the most frequent forwards actions, taken by at least 9 students each. Sixteen students take the first F-MT action, and nine (56%) of these take F-ADD and achieve the goal (taking the remaining steps shown) through forward-only actions. The 7 other students taking F-MT as a first step used a mixed approach to the problem.

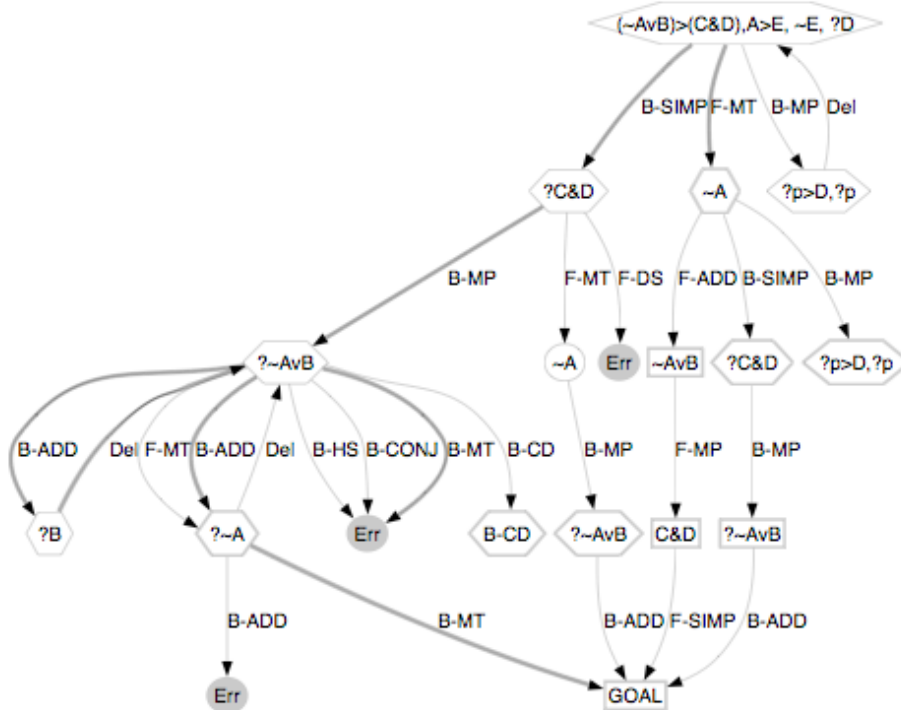


Figure 1. MDP of DT1.3 restricted to actions taken by at least 4 students.

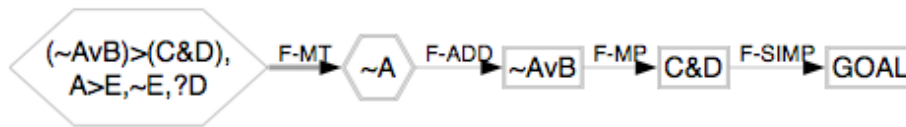


Figure 2, MDP restricted to frequent Forward actions, with at least 9 students taking each action.

Figure 3 shows the most frequent backwards actions, taken by at least 17 students each. Fifty-two students take the first B-SIMP action, and 45 of these take B-MP to derive $? \sim AvB$. Then, about half (24) of the students derive $?B$, which cannot be proven, and so delete this node and try again. A third of students (17) try using B-MT in error. Eventually, 41 students correctly derive $? \sim A$ and 35 of these reach the goal using backwards-only actions. Many of the remaining 17 students eventually succeed using a mixed approach, but this analysis shows a strong preference of students toward using backward-only approaches on this problem. This most likely reflects a focus on this approach in lectures, and also reflects the relative complexity of the premises in this problem. Although there are only 4 main actions that must be taken to solve this proof, we see that students do search to find correct approaches.

In looking at both backwards-only and forwards-only approaches, we see the most divergence for students in the steps connecting $\sim A$ with $\sim AvB$, using the addition rule (ADD). This suggests that students do not naturally think of applying addition. The first two authors have observed this anecdotally, but our visualization confirms this reluctance, and demonstrates that students attempt a number of other approaches before trying addition. Based on these findings, we can suggest more discussion of the addition rule in class, and can also plan to build hints into DT when addition applies. In the next section, we discuss how we plan to use the data generated by MDPs and the associated visualizations to target help to individual students using DT.

4. Generating Individualized Help

Assuming early identification of students at risk for failing the course, and more particularly, at risk for failing to master proof construction, the question becomes how



Figure 3. MDP restricted to frequent backwards actions, with at least 17 students taking each action.

to effectively address their needs. One alternative is to provide real-time, individualized hints to support on-going student proof construction efforts.

As we have proposed in [4], we plan to generate an MDP for each problem in DT and use it to generate hints for new students solving proofs in DT. Since DT has been used as a computer-aided instructional tool for a number of years, we have many semesters of data from which to create large MDPs for each DT problem. We plan to first use these MDPs to add intelligent hints to every problem. As a new student works a DT problem, we will match their states to those in the MDP. If their state is present in the problem's MDP, we will enable a Hint Button to give contextual help.

In [4], we have proposed several reward functions that could be used in hint generation. The three types of reward functions we have proposed are: 1) expert, 2) typical, and 3) least error-prone. The reward function we have described herein reflects an expert reward function, where the value for a state reflects the shortest path to the goal state. On the other hand, a "typical" reward function will choose a path through the MDP that reflects frequent student responses, giving high rewards to correct responses given by many students. The "least error-prone" function would assign high rewards to paths with low probabilities of errors. We plan to implement MDPs with each of these three rewards calculated for each state.

Given the current state, when the Hint Button is pressed, we will select a reward function for the current student based on their student profile. If we have identified the student as an at-risk student as described in section 3, we may select the "least error-prone" reward function for generating hints. On the other hand, high-performing students would likely benefit from expert hints, while students between these two extremes may benefit from hints reflecting typical student behavior [4].

After we've selected a reward function, we select the next state with the highest reward value. We propose three levels of hints from this state:

1. Tell the student what rule to apply next (rule hint).
2. Indicate the premises where the rule can be used (pointing hint).
3. Tell the student both the rule and the premises to combine, resulting in a "bottom-out" hint (e.g., giving the answer) [2].

We also propose to add a limit on the number of hints a student can use and still receive credit for working the problem. We believe that three hints is a fair amount, to be used on a single state in sequence as above or on separate states in the same problem. This results in giving the student one full step of the proof, or allowing rule hints up to three times.

If a student's state is not found in the MDP, the Hint Button will be disabled. Such a student can get DT's built-in feedback that indicates the correctness of each step, but will not get strategic help. However, we can add the student's action and its correctness to our database, and periodically run reinforcement learning to update the reward function values. Before an update is applied in DT, we will test the update to be sure that the instructors agree with the generated hints.

Once we have tested the feasibility of MDP-generated hints, we may group students according to their DT behavior and class performance, and run MDPs for each group of students. Then when the student asked for hints, the MDP chosen for that student will be tailored to both the current context (problem state), and characteristics of the student. For example, we foresee grouping students as those preferring to use backwards-only actions, forward-only, and mixed approaches, and those at-risk or not. Then, the suggested hints will be more likely to be usable by each student.

5. Conclusions and Directions for Future Research

We have proposed a two-pronged approach to using data to improve deductive logic education: 1) using prior course data to find early indicators of poor performance, and 2) deriving a way to leverage past student work in generating individualized help in writing proofs. In the first approach, we have determined that one applet (JT) is particularly important for student success in our Deductive Logic Course. Individualized feedback, reminding students to complete JT with at least 70% success, is one way to improve overall course performance. In the second approach, we have explored visualizations of student solutions to a logic proof in DT to determine other places for individualized help. We have concluded that even in a simple proof problem, there is a need for individualized help.

We have proposed an approach to generating these hints using both student characteristics and prior data. DT can already provide feedback on many errors students make. Adding MDPs to this tutor will enable it to provide individualized hints. These MDPs can constantly learn from new student data. We note that on cold start for a new problem that has no student data, the system will still act as a problem-solving environment, but after even one semester of data is collected, a limited amount of hints can be generated. As more data are added, more automated assistance can be generated. Once implemented, we will test the hints generated based on MDPs. We will investigate the effectiveness of 1) hints tailored according to a student's JT performance and general proof approaches, and 2) hints derived from expert, typical, and least error-prone MDPs. These resources promise to serve as a solid foundation for building our intelligent tutoring system.

References

- [1] J. Hartley and D. Sleeman, Towards more intelligent teaching systems. *International Journal of Man-Machine Studies* 2 (1973), 215-236.
- [2] K. VanLehn. The behavior of tutoring systems, *International Journal of Artificial Intelligence in Education* 16 (2006), 227-265.
- [3] T. Murray, Authoring intelligent tutoring systems: An analysis of the state of the art, *International Journal of Artificial Intelligence in Education* 10 (1999), 98-129.
- [4] T. Barnes and J. Stamper, Toward the extraction of production rules for solving logic proofs, *Proc. 13th Intl. Conf. on Artificial Intelligence in Education, Educational Data Mining Workshop (AIED2007)*, Marina del Rey, CA, July 9, 2007.
- [5] R.E. Mayer, *Thinking, Problem Solving, Cognition*, Freeman, New York, 1992.
- [6] M.J. Croy, Graphic interface design and deductive proof construction, *Journal of Computers in Mathematics and Science Teaching* 18 (1999), 371-386.
- [7] M.J. Croy, Problem solving, working backwards, and graphic proof representation, *Teaching Philosophy* 23 (2000), 169-187.
- [8] R. Scheines and W. Sieg, Computer environments for proof construction, *Interactive Learning Environments* 4 (1994), 159-169.
- [9] E. F. Schumacher, *Small is Beautiful*, Harper and Row, New York, 1973, 89.
- [10] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA.