Evaluation of Logic Proof Problem Difficulty through Student Performance Data

Behrooz Mostafavi
North Carolina State University
Department of Computer Science Raleigh, NC 27695
bzmostaf@ncsu.edu

Tiffany Barnes
North Carolina State University
Department of Computer Science Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT

The interactions of concepts and problem-solving techniques needed to solve open-ended proof problems are varied, making it difficult to select problems that improve individual student performance. We have developed a system of data-driven ordered problem selection for Deep Thought, a logic proof tutor. The problem selection system presents problem sets of expert-determined higher or lower difficulty to students based on their measured proof solving proficiency in the tutor. Initial results indicate the system improves student-tutor scores; however, we wish to evaluate problem set difficulty through analysis of student performance to validate the expert-authored problem sets.

Keywords

Problem Difficulty, Logic Proof, Data-driven Problem Selection

1. INTRODUCTION

Effective intelligent tutoring systems present problems to students in their zone of proximal development through scaffolding of major concepts [3]. In domains such as deductive logic, where the problem space is open-ended and requires multiple steps and knowledge of different rules, it is difficult to choose problems for individual students that are appropriate for their proof-solving ability. We have developed a system that uses the data-driven knowledge tracing (DKT) of domain concepts in existing student-tutor performance data to regularly evaluate current student proficiency of the subject matter and select successive structured problem sets of expert-determined higher or lower difficulty.

We used an existing proof-solving tool called Deep Thought to test the DKT problem selection system. The system was integrated into Deep Thought and tested on a class of undergraduate philosophy students who used the tutor as assigned homework over a 15-week semester. Performance data from this experiment were compared to data from previous use of Deep Thought without the DKT problem selection system. The results of the comparison indicate that the DKT problem selection system is effective in improving student-tutor performance. However, we wish to evaluate the difficulty of presented problems using student performance data to validate the difficulty of expert-determined problem sets, and improve the system for future students.

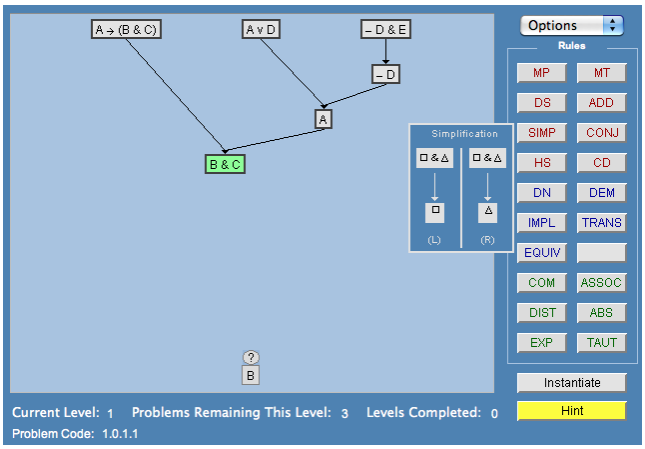

Figure 1: A screen capture of the Deep Thought tutor, showing given premises at the top, conclusion at the bottom, and rules for application on the right.

2. DEEP THOUGHT

Fig. 1 shows the interface for Deep Thought, a web-based proof construction tool created by Croy as a tool for proof construction assignments [1]. Deep Thought displays logical premises, buttons for logical rules, and a logical conclusion to be derived. For example, the proof in Fig. 1 provides premises $A \rightarrow (B \wedge C)$; $A \vee D$; and $\neg D \wedge E$, from which the user is asked to derive conclusion B using the rules on the right side of the display window.

Deep Thought keeps track of student performance for the purpose of proficiency evaluation and post-hoc analysis. As a student works through a problem, each step is logged in a database that records: the current problem; the current state of progress in the proof; any rule applied to selected premises; any premises deleted; errors made (such as illegal rule applications); completion of the problem; time taken

per step; elapsed problem time; knowledge tracing scores for each logic rule in the tutor.

2.1 Problem Selection

The problem selection system in Deep Thought presents ordered problem sets to ensure consistent, directed practice using increasingly related and difficult concepts. The system presents set of problems at different degrees of difficulty, determined through evaluation of current student performance in the tutor.

Evaluation of student performance is performed at the beginning of each level of problems. Level 1 of Deep Thought contains three problems common to all students who use the tutor, and provides initial performance data to the problem selection model. Levels 2–6 of Deep Thought are each split into two distinct sets of problems, labeled higher and lower proficiency. The problems in the different proficiency sets are conceptually identical to each other, prioritizing rules important for solving the problems in that level. To prevent students from getting stuck on a specific proof problem, Deep Thought allows students to temporarily skip problems within a level. A unique case occurs if a student skips a problem more than once in a higher proficiency problem set; the student will be dropped to the lower proficiency problem set in the same level, under the assumption that the student was improperly assigned the higher proficiency set (See Fig. 2).

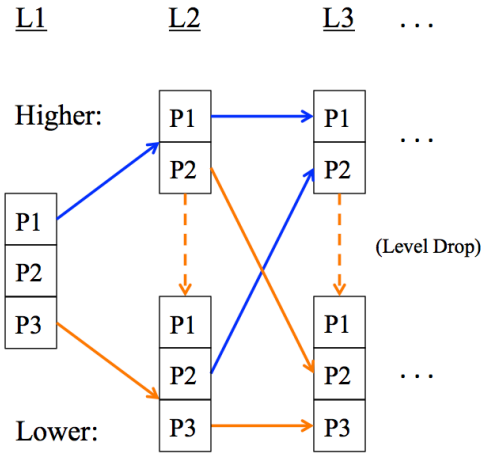

Figure 2: DT2 path progression. At each level, students are evaluated and provided either the higher or lower proficiency problem sets. Students can also be switched from the higher to lower proficiency set within a level.

2.2 Logic Proof Problems

The degree of problem solving difficulty between proficiency sets is different, as determined by domain experts. The problems in the low proficiency set require fewer numbers of steps for completion, lower complexity of logical expressions, and lower degree of rule application than problems in the high proficiency set (See Table 1).

3. DATA GRAPH REPRESENTATION

Table 1: An example of lower and higher proficiency set problems from Deep Thought requiring the same concepts: Level 4 Problem 3 from the lower proficiency set (top); Level 4 Problem 2 from the higher proficiency set (bottom). The prioritized rules required for these problems are Conjunction and Constructive Dilemma.

#	*Premise*	*Derivation*
1	$(A \rightarrow B) \wedge (\neg D \rightarrow F)$	Given
2	$A \vee \neg D$	Given
3	$\neg A \rightarrow (D \vee G)$	Given
4	$\neg A$	Given
5	$B \vee F$	1,2/Constructive Dilemma
6	$\neg D$	2,4/Disjunctive Syllogism
7	$D \vee G$	3,4/*Modus Ponens*
8	G	6,7/Disjunctive Syllogism
9	$(B \vee F) \wedge G$	5,8/Conjunction

#	*Premise*	*Derivation*
1	$Z \rightarrow (\neg Y \rightarrow X)$	Given
2	$Z \wedge \neg W$	Given
3	$W \vee (T \rightarrow S)$	Given
4	$\neg Y \vee T$	Given
5	Z	2/Simplification
6	$\neg W$	2/Simplification
7	$\neg Y \rightarrow X$	1,5/*Modus Ponens*
8	$T \rightarrow S$	3,6/Disjunctive Syllogism
9	$(\neg Y \rightarrow X) \wedge (T \rightarrow S)$	7,8/Conjunction
10	$X \vee S$	4,9/Constructive Dilemma

Deep Thought was used as a mandatory homework assignment by students in a philosophy deductive logic course ($n = 47$). Students were allowed to work through the problem sets at their own pace for the entire 15-week semester. Problem Levels 1–6 were assigned for full completion of the tutor, totaling 13–18 (out of the total tutor-set of 43) problems depending on proficiency path progression.

For the purpose of problem difficulty evaluation, progress through the tutor can be expressed as a directed graph for each individual student, with nodes in the graph each corresponding to a single problem. The node set for the graph represents the problem space for the tutor, and is the same for every student. Each problem node has the following properties:

1. Tutor Level (1–6)
2. Proficiency (High or Low)
3. Problem Number (1–3)
4. Problem Complete (True or False)
5. Expert-Authored
   (a) Required Rules
   (b) Minimal Solution
6. Corresponding Step Logs (See Section 2)

Directed edges between nodes correspond to movement between problems by the individual student, and are assigned a numerical value, ordered by increasing time stamp. The nodes and directed edges together give a map of the student's progression through the tutor. Connected nodes with false Problem Complete status represent a skipped problem, and the node adjacent to the highest numbered edge represents the student terminus point in the tutor. Isolated nodes represent non-visited problems, and are therefore un-useable for problem difficulty evaluation.

Logic proofs can also be represented as directed graphs, with each node containing a proof premise, and each directed edge indicating a node parent-child relationship, along with an applied logic rule. For example, the top proof shown in Table 1 can be represented as a graph with the premise in each line as a node, with the directed edges into that node corresponding to the derivation of that premise from parent nodes. A proof premise can either be a variable (i.e. A), a negated variable or expression (i.e. $\neg A$, or $\neg(A \wedge B)$), or an operational expression in (variable/nested expression)-operand-(variable/nested expression) form (i.e. $A \vee B$, or $(A \wedge B) \vee (A \rightarrow B)$). Nested expressions can be represented in high level form. Therefore, node premises can be categorized by their operand (conjunction, disjunction, negation, implication, equivalence), the complexity of the expression (single variable, simple expression, complex [nested] expression), and the rule used for derivation.

4. PROBLEM DIFFICULTY EVALUATION

The question at hand is how to best use the recorded data to determine proof problem difficulty through student performance. We wish to find both a classification of problem difficulty between proficiency sets in the same level, and difficulty of all problems in the tutor, compared to expert-determined classifications.

Because students follow different problem-solving paths, no student can solve all available problems in the tutor, nor are students likely to solve problems in both proficiency sets within the same level. This makes student performance comparison over multiple problems difficult. We plan to use a combination proof-problem properties weighted by student performance metrics to evaluate problem difficulty; however, we have not determined which combination of methods to use. We are currently looking into weighted cluster-based classification methods to apply to the problems. The hypothesis presumed before applying one of these methods would be that problems of similar difficulty would be placed into the same clusters. Student performance metrics for each problem could be used to determine distance, since it's assumed that students would react most similarly to problems of similar difficulty. Eagle et al. applied network community mining to this student log data in order to form interaction networks [2]; a modified version could be applied here on a student-per-problem level in order to determine prominent similar behaviors that are correlated with problem performance.

This would determine which problems are of similar difficulty, but not necessarily which problems (or groups of problems) are more or less difficult. That determination could be made by analyzing student rule scores across problems, or even the difference in scores at the start and end of a problem. In particular, analyzing the difference in rule scores would both standardize the scores (to account for the scores being calculated at different points in the tutor) and give a measure of forward or backward progress (a student's rule scores should not decrease after solving an easy problem).

Problem properties we feel are valuable to take into consideration when evaluating problem difficulty per student include:

- Classification of problems by operand/expressions
- Deviation of student solutions from expert solutions
  - Number of steps taken
  - Number and frequency of rules used

Student performance metrics that we feel are valuable to take into consideration include:

- Path progression through the tutor, including
  - Order of assigned proficiency sets
  - Number and path location of skipped problems
  - Terminus point in tutor
  - Final tutor grade
- Knowledge tracing scores for each rule, prioritized by problem requirements
- Step and elapsed time
- Type and number of errors committed

We would appreciate any literature recommendations, as well as suggestions for how to use the data from our experiment to measure and compare problem difficulty through student performance.

5. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. 0845997.

6. REFERENCES

[1] M. J. Croy, T. Barnes, and J. Stamper. Towards an Intelligent Tutoring System for Propositional Proof Construction. In *Current Issues in Computing and Philosophy*, pages 145 – 155. 2008.

[2] M. Eagle, M. Johnson, and T. Barnes. Interaction Networks: Generating High Level Hints Based on Network Community Clusterings. In *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)*, pages 164–167, 2012.

[3] T. Murray and I. Arroyo. Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (EDM 2002)*, pages 289 – 294, 2002.