Automatic Generation of Proof Problems in Deductive Logic

B. MOSTAFAVI, T. BARNES, AND M. CROY
University of North Carolina at Charlotte, United States

Automatic problem generation for learning tools can provide the required quantity and variation of problems necessary for an intelligent tutoring system. However, this requires an understanding of problem difficulty and corresponding features of student performance. Our goal is to automatically generate new proof problems in Deep Thought – an online propositional logic learning tool – for individual students based on their performance and a set of instructor parameters.. In an initial exploratory study, we evaluate the generated problems compared to the original set of stored problems. This evaluation uses collected student data and instructor feedback.

Key Words and Phrases: Logic proof, problem generation, problem difficulty, intelligent tutoring system

1. INTRODUCTION AND RELATED WORK

Intelligent tutoring systems (ITS) provide adaptive instruction to students, and have a significant effect on learning [Murray, 1999]. Much of the work in ITS development has been in generating feedback and hints for existing problems, such as CTAT, an example-based authoring tool [Koedinger et al, 2004]. Logic-based tutors such as Logic-ITA support the learning and teaching of logic proofs, verifying proof statements, providing feedback, and logging data for exploration [Lesta and Yacef, 2004] [Yacef, 2005].

Our focus, however, is on automatic problem generation. We present our work in the context of a logic proof tutor, called Deep Thought [Croy, Barnes and Stamper, 2008]. In Deep Thought, students construct proofs by applying logical rules to a set of given premises in order to generate a specific conclusion. Our long-term goal is to provide instantly generated proof problems that are appropriate based on the student's skill level, course progression, and previous performance on Deep Thought problems.

McGough et al [2001] created a web-based dynamic problem generation system in engineering; however, their problems are assembled from a pool of existing problem subsets, while we seek to generate logic proofs from scratch. Beck et al [1997] accomplished this for a tutor in arithmetic operations, which is similar to what we wish to accomplish using logic proofs. However, that tutor took into account several assumptions about a student's basis of knowledge in basic arithmetic, which are not as well defined with logic proof construction.

2. THE AUTOMATIC PROBLEM GENERATOR

Deep Thought is an existing web-based tool with a graphical user interface that provides a set of problems that display logical premises, buttons for logical rules (axioms), and a conclusion that a student must reach by applying logical rules to the premises (Figure 1). Student progress is logged at each step of proof construction, recording attributes including rule use, errors, deletions, time, and successful completion of the problem.

Authors' addresses: B. Mostafavi, Department of Computer Science, University of North Carolina at Charlotte, United States. E-mail: bzmostaf@uncc.edu; T. Barnes, Department of Computer Science, University of North Carolina at Charlotte, University of North Carolina at Charlotte; E-mail: Tiffany.Barnes@uncc.edu; M. Croy, Department of Philosophy, University of North Carolina at Charlotte, United States. E-mail: mjcroy@uncc.edu

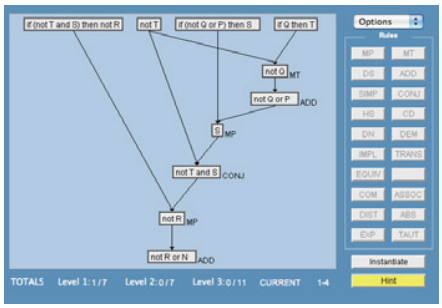

Figure 1. Deep Thought user interface, showing a successfully completed problem (English mode).

We have developed a java-based background process to Deep Thought called LQGen (Logic Question Generator) that automatically generates proof problems that satisfy the conceptual requirements of the course instructor. LQGen takes as input the parameters of the desired problem, and generates a new random problem. The input parameters are extracted from the expert-derived solution of the original problems, and include the number of premises, re-used premises, logical rule use, number of steps, and complexity of statements and conclusion (see Table 1).

LQGen generates problems by working backwards [Croy, 2000], starting with the problem conclusion to generate subsequent steps until a full problem is developed. After creating a random conclusion, it builds a tree of logical statements to a depth equal to the number of steps in the expert solution, based on the parameter requirements, then traverses the tree and deletes branches until the specified number of premises is reached.

In its current state, LQGen generates problems that match the parameters of the expert-solved original problems. However, the parameters do not adapt to an individual student's skill level and performance. Before we can accomplish this, we must first understand how students might behave in constructing logic proofs by examining the features that might determine a problem's difficulty, and compare this behavior between original Deep Thought problems and problems generated by LQGen.

3. MEASURING PROBLEM DIFFICULTY

If we accept the general idea that certain logical concepts are more difficult than others, we can assume that the parameters we use for construction of proofs may be sufficient for generation of problems that have the same conceptual difficulty. However, we need to look at student data to determine how student performance indicates their level of knowledge, and therefore problem difficulty. Beck and colleagues [1997] look at historical student data to determine the rate at which an arithmetic concept is learned, and how often a student continues to use the concept. However, in their tutor, the knowledge gained by the students and the method of proficiency demonstration is linear, which is not the case with logic proofs, as students can solve problems in various ways.

There are several factors we can consider in determining problem difficulty. We can consider the failure rate in the attempts made by the students per problem, and the usage of rules as expected from the expert solution. If a student has a decreasing rate of failure and higher usage of rules for problems of similar type, we can assume the student is performing at or near the level of difficulty set by the course instructors. We can also look at the number of performance errors, any reworking of the problem, the number of rule applications, and the elapsed time. We would expect an exponential learning curve in the factors above when students are working on problems with similar concepts.

Once we determine how these factors affect student performance from the student data, we can determine how to alter the parameters of the problems created by LQGen to give students practice and mastery of the concepts taught before generating problems of higher difficulty.

4. METHODS AND RESULTS

Data are from students solving Deep Thought Level 1 problems as homework in in a Deductive Logic course. Eighty Fall 2010 students solved instructor-authored problems, and eighty Spring 2011 students solved LQGen problems designed to match the Fall problems. Table 1 shows the parameters of the six problems as solved by an expert, using inference rules including modus ponens (MP), disjunctive syllogism (DS), simplification (SIMP), hypothetical syllogism (HS), modus tollens (MT), addition (ADD), conjunction (CONJ), and constructive dilemma (CD). The problem set was designed so that students would use all of the inference rules over the set. We expected problem 4 to be more difficult since it is the first to use CONJ, and it requires the re-use of a given premise. Problem 5 was difficult for the same reasons plus it also introduced CD.

Table 1. Deep Thought Level 1 Parameters as determined by expert solutions

Problem	Premises	# Rules Used	Rules used by expert								Premise Reuse
			MP	DS	SIMP	HS	MT	ADD	CONJ	CD	
1.1	3	4	1	1	2	0	0	0	0	0	No
1.2	3	4	1	1	1	0	0	1	0	0	No
1.3	3	4	1	0	1	0	1	1	0	0	No
1.4	4	6	2	0	0	0	1	2	1*	0	Yes*
1.5	4	6	1	1	2	0	0	0	1	1*	Yes*
1.6	4	4	1	1	1	1	0	0	0	0	No

Figure 2 shows the percent of incomplete (failed) attempts, with a learning curve decreasing from problems 1.1 to 1.3 and then again from 1.4 to 1.6, consistent for both original and generated problems.

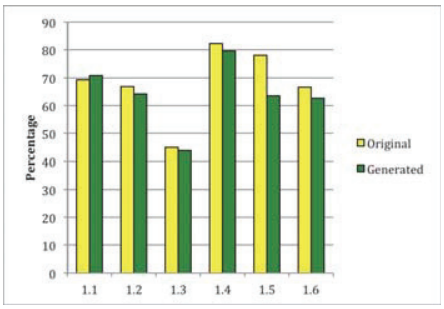

Figure 2. Average percent of incomplete attempts for Deep Thought Level 1 problems

Figure 3 shows that the number of attempts for each problem has a reasonable learning curve from 1.1-1.3 and 1.4-1.6 (with low attempts for 1.2 and 1.3 since they were not required). Figure 4 shows the average time taken per problem attempt. The generated problem 1.4 has more attempts and more time per attempt, showing that it is likely more difficult than the original problem, while generated 1.5 has fewer attempts and much shorter times. One explanation for fewer and shorter tries on 1.5 is that after solving the harder generated 1.4 problem, students had less trouble with 1.5. However, upon investigation, we also found that while both 1.5 problems needed constructive dilemma (CD), the original 1.5 problem needed it much later in the problem so students had to determine how to set up the problem to apply CD, while in the generated problem,

it was apparent to the students that CD could be applied directly to the premises. The number of attempts is greater for generated problems in 4 of the 6 problems, and on the remaining two the number of attempts is very similar. This suggests that the Spring students attempted the generated problems more times, but spent less time on each attempt, than the Fall students tried the original problems.

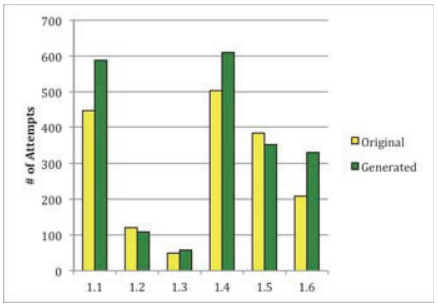

Figure 3: Total attempts (1.2 & 1.3 optional)

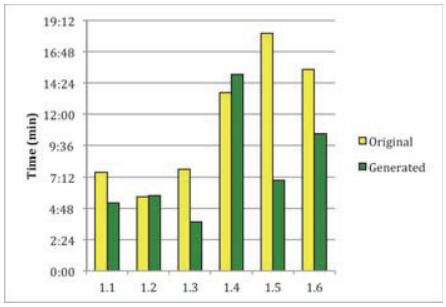

Figure 4. Average time taken per attempt

At first we expected that spring students, who had more short attempts, realized when an attempt was not productive and simply started over, but our analysis of problem length suggests otherwise. Figure 5 shows the total number of errors, deletions, and steps in student attempts. All the generated problems have more steps, therefore students reworked the generated problems more often, with longer proofs, but in shorter average times, than students did with the original problems. We believe this suggests that the generated problems needed longer time, but were not quite as difficult proofs.

Although students spent more time on each attempt for generated problems, they performed slightly more deletions on 1.1, 1.2, and 1.4 and made more errors for 1.2, 1.3, 1.4, and 1.6. This may indicate that students in Spring 2011 are finding rule applications easier and can apply them in Deep Thought more quickly per step, and may be spending more time in constructing proofs and trying more strategies in the tutor.

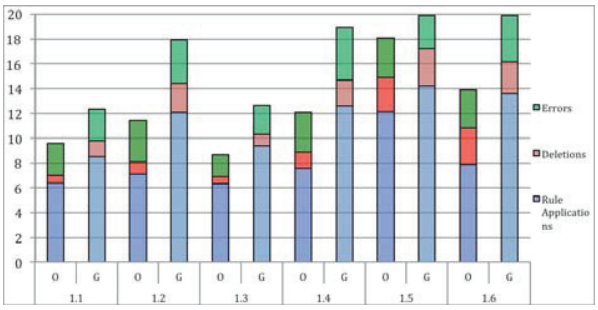

Figure 5. Average number of errors, deletions, and rule applications per attempt for original (O) and generated (G) Deep Thought Level 1 problems

Our expert took 4 - 6 steps on every original and generated problem, but as expected, students applied more rules on average per attempt for both problem sets, as is particularly apparent in Table 2. For problem 1.1 students replaced a use of the SIMP rule with MP in LQGen. LQGen's problem 1.2 seemed to require one more application of DS and ADD for students, but problem 1.4 didn't need MP and MT as much as the original

problem. Students used SIMP a bit more in the LQGen 1.4 problem, and solved LQGen 1.6 with one less MP and HS rule each. Overall, both original and generated problems encouraged students to apply the rules experts used, and the numbers of times students applied the rules in correct solutions across the whole problem set shows a similar pattern to their usage by experts (though each rule is used more by students).

Table 2. Average rule use per successfully completed problem for Deep Thought Level 1

		MP	DS	SIMP	HS	MT	ADD	CONJ	CD
1.1	*Expert*	*1*	*1*	*2*					
	Orig.	1.73	1.78	**2.5**	.30	.06	.34	.33	.08
	Gen.	2.11	1.02	1.25	.26	.06	.44	.21	.16
1.2	*Expert*	*1*	*1*	*1*			*1*		
	Orig.	1.83	1.78	2.18	.72		1.6	.32	.20
	Gen.	1.76	**2.61**	2.5	.17	.12	**2.46**	.56	.10
1.3	*Expert*	*2*		*1*		*1*	*1*		
	Orig.	1.37	.22	1.52	.07	1.22	1.29	.14	.06
	Gen.	1.34	.09	1.37		1.25	1.53	.09	.12
1.4	*Expert*	2				1	1	1	
	Orig.	**2.02**	.45	.63	.84	**2.03**	2.32	2.04	.31
	Gen.	1.15	.75	.99	.72	1.44	**2.90**	2.18	.48
1.5	*Expert*	*1*	*1*	*2*				*1*	*1*
	Orig.	**2.17**	2.38	2.45	.29	.29	.94	1.35	.95
	Gen.	1.22	2.15	2.85	.32	**.63**	.85	1.57	.85
1.6	*Expert*	*1*	*1*	*1*	*1*				
	Orig.	**2.07**	2.02	2.1	**2.32**	.81	.12	.42	.01
	Gen.	1.34	2.03	2.36	1.34	.85	.48	.48	.21

		MP	DS	SIMP	HS	MT	ADD	CONJ	CD
Total	*Expert*	*7*	*4*	*7*	*1*	*2*	*3*	*2*	*1*
	Orig.	*11.2*	*8.6*	*11.1*	*4.5*	*4.4*	*6.6*	*4.6*	*1.6*
	Gen.	*8.9*	*8.6*	*11.3*	*2.8*	*4.4*	*8.7*	*5.1*	*1.9*

5. CONCLUSION AND FUTURE WORK

We expected that, using parameters from expert solutions to logic proof problems, we could develop LQGen to generate similar problems for students to practice in deductive logic. Through the features compared in the above section, we believe that we have shown that LQGen can be used to provide practice problems that target a given rule set.

We plan to continue developing LQGen to support all the logical rules available in Deep Thought. We also plan to combine LQGen with a student model to assign problems that will encourage mastery of each rule needed to solve proofs. Once LQGen has been updated using knowledge gained from the study and integration of adaptive difficulty settings, it will be fully integrated into Deep Thought for its use in course instruction.

ACKNOWLEDGEMENTS

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

REFERENCES

BECK, J., STERN, M., AND WOOLF, B.P. 1997. Using the Student Model to Control Problem Difficulty. In Jameson, A., Paris, C., and Tasso, C. (Eds.) *User Modeling: UM97 Proceedings*, Springer, New York.

CROY, M., BARNES, T., AND STAMPER, J. 2008. Towards an Intelligent Tutoring System for Propositional Proof Construction. In Briggle, A., Waelbers, K., and Brey, E. (Eds.) *Current Issues in Computing and Philosophy*, 145-155, IOS Press, Amsterdam, Netherlands.

CROY, M. 2000. Problem Solving, Working Backwards, and Graphic Proof Representation. *Teaching Philosophy*, 23, 169-187.

KOEDINGER, K., ALEVEN, V., HEFFERNAN, T., MCLAREN, B., AND HOCKENBERRY, M. 2004. Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. In *ITS 2004 Proceedings*, 162-173.

LESTA, L. AND YACEF, K. 2004. An Intelligent Teaching Assistant System for Logic. In *ITS 2004 Proceedings*, 421-431.

MCGOUGH, J., MORTENSEN, J., JOHNSON, J., AND FADALI, S. 2001. A Web-Based Testing System with Dynamic Question Generation. In *FIE 2001 Proceedings*, 3, 23-28.

MURRAY, T. 1999. Authoring Intelligent Tutoring Systems. In *IJAIED,* 10, 98-129.

YACEF, K. 2005. The Logic-ITA in the Classroom: A Medium Scale Experiment. In *IJAIED,* 15, 41-62.