Using Constraints to Develop and Deliver Adaptive Tests

Sophiana Chua Abdullah and Roger E. Cooley
Computing Laboratory
University of Kent at Canterbury
Canterbury
Kent CT2 7NF
The United Kingdom
Email: {sc34,rec}@ukc.ac.uk
Telephone: ++44 1227 823824 Fax: ++44 1227 762811

Abstract

This paper shows how the techniques used to develop knowledge-based systems can be applied to the construction of adaptive tests. It reports on an experimental approach to the construction of adaptive tests, and it draws on work in the fields of Intelligent Tutoring Systems, Expert Systems and Constraint Logic Programming.

The distinctive features of the work are:

- The use of expert emulation as a basis for the design of tests.
- The use of logic programming and constraints for two purposes. Firstly, to support knowledge acquisition from an expert tutor during the development of a test; and secondly, to support the delivery of the test.

The paper, after reviewing the approach to adaptive tests in Intelligent Tutoring Systems research, presents a case for expert emulation under those circumstances in which statistically based testing procedures are inappropriate. The paper then describes how knowledge about the content and design of an adaptive test can be facilitated by computer support. The software developed for this task makes use of a constraint solver embedded in a Prolog system. In the subject area of the adaptive tests, namely elementary arithmetic, it is shown that this software can be used for several purposes. It can be used to: describe classes of problems, describe arithmetic skills, describe student responses to problems and to generate problems. The paper concludes with a discussion of a procedure for exploring a student's knowledge of a syllabus.

Adaptive Testing in Intelligent Tutoring Systems

Though developed independently, computerised adaptive testing (Wainer, 1990) has found a natural home in intelligent tutoring systems. In particular, ideas from Item Response Theory (Wainer and Mislevy, 1990) have been very influential. They have formed the basis of a system to assess student programming abilities (Syang and Dale, 1993), they have influenced Huang's content-balanced tests (Huang, 1996), and more recently, their influence can be seen in a web-based adaptive testing system in the domain of European vegetable species (Rios *et al.*, 1999). Another strand of development in adaptive testing is concerned with describing the structure of a body of knowledge. This has given rise to the work of the ARIES laboratory at the

University of Saskatchewan. Here, subject domains such as mathematics have been represented using the notions of "granularity hierarchies" and "prerequisite relationships". This work is also distinguished by the use of a Bayesian approach to probability to make inferences about students' states of knowledge so as to optimise the adaptive testing process (Collins *et al.*, 1996). Other work, primarily distinguished by its concern with subject domain representation, has been carried out by Falmagne, Doignon, Koppen, Villano, and Johannesen, who use the idea of "knowledge spaces" (Falmagne *et al.*, 1990), and Dowling, Hockemeyer, and Ludwig who augment the idea of "knowledge spaces" with the concept of "fringes" (Dowling *et al.*, 1996).

The concern of these researchers with representation is motivated by the desire to automate the progression of a test from one topic to another, and from one level of complexity to another. In tackling this task, the problem of the justification for any structuring of a syllabus is ignored. Though there may be, from some given point of view, an optimal way of structuring a syllabus, the view adopted in this research is that it is a subjective matter to be determined by an expert teacher. Studies of intelligent tutoring systems have shown that, as one would expect, it is difficult to transfer systems from one setting to another, because there is considerable cultural variation in both teaching and learning (Payne and Squibb, 1990). This provides the prime motive for investigating techniques based on expert emulation for the production of test for local consumption.

This technique has an additional advantage. A lack of homogeneity amongst a student body can weaken the effectiveness of techniques based on population statistics; and the target body of students with which this paper has been concerned are, educationally, not very homogeneous. It is the rather transient group of adult prisoners who opt for remedial help with elementary arithmetic.

A Strategy for Knowledge Acquisition

There are several problems to be confronted when adopting an expert emulation approach to designing an adaptive test. They include the problem of finding suitable experts (Lightfoot, 1999), and they are compounded by the possibility that the objectives of a computer delivered adaptive test being quite distinct from the objectives of a more conventionally delivered test. In addition, there is a need for knowledge representation: a need for a method of representing classes of problems that are to form the contents of an adaptive test. Previous work on knowledge acquisition has been carried out by Dowling and Kaluscha (1995), and Khuwaja (1996) has discussed acquisition for general use in the development of intelligent tutoring systems.

The approach to knowledge acquisition in the research described here is to separate the task of designing an adaptive test into the following sub-tasks:

- describing classes of problems,
- describing skills,
- describing responses to problems,
- problem generation,
- problem progression.

Software has been developed to support these subtasks using Constraint Logic Programming.

A knowledge acquisition exercise has been carried out to evaluate our approach. The tutor, who is being studied, teaches mathematics to male adults who are serving time at a local prison. He aims to enable these prisoners to obtain qualifications from the Associated Examining Board or the City & Guilds body. The tutor frequently finds it necessary to evaluate new students. Interestingly, although he uses some standardised pencil-and-paper tests, he also uses a manual adaptive testing procedure. By means of techniques of structured interviewing and task analysis (McGraw and Harbison-Briggs, 1989), the strategy of the tutor in developing an adaptive test was elicited. This exercise involved approximately 20 hours of interviews spread over a period of three months. The exercise was concerned exclusively with the addition of fractions.

Constraint Logic Programming for Knowledge Acquisition

The history and background of constraint programming is usefully summarised by Marriott and Stuckey (1998). It has been used in many real life problems such as airline scheduling and container port scheduling (Abbott, 1995). It is concerned with the use of constraints to simplify the solution of search problems. Constraint programming modules are available for a range of programming platforms. The work presented below uses the notation of clp(FD), "constraint logic programming over finite domains" (Carlsson *et al.*, 1997), which is integrated with SICStus Prolog. (A commercially available Prolog systems developed and distributed by the Swedish Institute of Computer Science). Constraint logic programming provides a language for the description of relationships in the form of constraints and a mechanism to calculate a set of values which satisfy those constraints.

Clp(FD) was actively used by the interviewer conducting knowledge acquisition interviews. The teacher, who is the target of the emulation, is of course, not expected to write constraints, but is more than likely to take an interest in them. During discussions, which involve the production of example problems, the interviewer enters the necessary constraints, or modifies existing constraints, to describe the particular class of problem under discussion. The set of constraints is then solved interactively to produce example problems. These allowed the interviewer to obtain confirmation of what had been elicited and formed the basis of further rounds of discussion and modification. For most classes of problems, it is not feasible to expect the teacher to inspect every example. This means that unexpected and undesirable examples may be not be revealed during this knowledge acquisition process. Traditional program testing and additional knowledge acquisition sessions are needed to reduce the probability of errors.

Using clp(FD), domains can be enlarged or restricted, and constraints can be added or removed. Easy problems may be characterised by the use of single digit integers, harder problems may be those in which the use of specific skills is necessary. As with all knowledge acquisition, good preparation by the interviewer is extremely valuable. This can conveniently take the form of

developing some speculative constraints, but this should not be allowed to influence the interviews. The aim of emulating the teacher must be paramount.

Describing Problem Classes

One of the first tasks of the tutor is to identify an area of curriculum or topic that is to be tested. The next task for computerised adaptive testing systems based on the Item Response Theory is usually the construction of test questions or "items" for an item bank (Linacre, 1995). The strategy adopted by our tutor is somewhat different. He has chosen to categorise problems into several classes:

- Add two proper fractions of common denominators
- Add a proper fraction and an improper fraction of common denominators
- Add two improper fractions of common denominators
- Add two proper fractions of different denominators
- Add a proper fraction and an improper fraction of different denominators
- Add two improper fractions of different denominators

The description of a class of problems is treated as a set of constraints and this consists of a set of variables, a statement of the domains of the variables, and a statement of the relational constraints that hold between the variables. For example, during an interview, the tutor wanted to represent a class of problems, which involved the addition of two proper fractions with a common denominator of the form,

$$\frac{N1}{D1}+\frac{N2}{D2}=\frac{N}{D}$$

and he wanted to use single digit integers.

This can be represented in clp(FD) as a code fragment:

```
domain([N1,D1,N2,D2],1,9),      % Single digit integers
N1 #<  D1,                      % First operand - proper fraction
N2 #<  D2,                      % Second operand - proper fraction
D1 #= D2.                       % A common denominator
```

Describing Skills

The skills involved in solving problems in addition of fractions were identified by the tutor as:

- Add equivalent fractions
- Cancel fraction
- Make proper
- Find the lowest common multiple
- Find equivalent fractions

These can be represented in clp(FD). For example, the *cancel fraction* skill can be represented as:

```
% Simplify the fraction N/D into its lowest form to give X/Y
% Example:  63/81 gives 7/9
cancel_fraction(N,D,X,Y) :-
        domain([N,D,X,Y,F], 1,99),
        F*X #= N,
        F*Y #= D,
        maximize(labeling([], F,X,Y],), F).
```

Here, variable F is the common factor to be cancelled. This is specified by the two relational constraints. The *maximize* predicate in the final line ensures that the largest value of F will be found.

Describing Responses to Problems

An adaptive test might only be concerned with the mathematical correctness of a student's response to a problem. The following possible answer types have been identified during knowledge elicitation:

- Proper fraction in its simplest form (e.g. 1/2)
- Whole number = 1 (e.g. 3/3)
- Proper fraction which can be simplified further (e.g. 6/8)
- Improper fraction in its simplest form (e.g. 4/3)
- Improper fraction which can be simplified further (e.g. 10/6)
- Whole number > 1 (e.g. 8/2)

The use of clp(FD) has some advantages when emulation of a teacher requires a diagnostic approach to characterising students' answers. The non-procedural nature of constraint programming results in it being just as easy to check the values of a set of variables, as it is to generate them. For example, if the intention were to specify a problem with a result that is a proper fraction, this could be achieved with the addition of another constraint:

```
N #< D
```

where N and D can take any integer value from 1 to 99.

These constraints can be added to the previous code fragment, thus:

```
domain([N1,D1,N2,D2],1,9), % Single digit integers
domain([N,D], 1, 99),      % Possible values for the answer
N1 #<  D1,                 % First operand - proper fraction
N2 #<  D2,                 % Second operand - proper fraction
D1 #= D2,                  % A common denominator
N #< D.                    % Answer must be a proper fraction
```

Likewise, if the intention was to have a result that is an improper fraction, the constraint N #< D can be replaced by N #> D.

Problem Generation

Problems can be generated “on the fly” from placing constraints on what the tutor wants: problem type, response type and skills used. This is in contrast to maintaining large test item banks commonly associated with adaptive testing systems based on item response theory.

For example, a problem can be generated by specifying the problem class and the required response type. Once the description of a class of problems and their appropriate responses is treated as a set of constraints, it must be satisfied by every example of that class. This is achieved by the *labeling* predicate which will initiate a search for solutions for all the variables in the arithmetic expression,

N1/D1 + N2/D2 =:= N/D.

For instance, if we want to a problem where two proper fractions are added to give another proper fraction, this can be specified as follows:

```
problem_class(N1,D1,N2,D2,N,D) :-
        domain([N1,D1,N2,D2],1,9),          % Single digit integers
        domain([N,D], 1, 99),               % Possible values for the answer
        N1 #<  D1,                          % First operand - proper fraction
        N2 #<  D2,                          % Second operand - proper fraction
        D1 #= D2,                           % A common denominator
        N #< D,                             % Must be a proper fraction
        labeling( [], [N1,D1,N2,D2,N,D]),   % Generate values for variables
        N1/D1 + N2/D2 =:= N/D.              % The arithmetic expression
```

A solution from the execution of the above code is:

N1 = 1, D1 = 3, N2 = 1, D2 = 3, N = 2, D = 3

that is, the generation of a problem, $\frac{1}{3}+\frac{1}{3}=\frac{2}{3}$, which satisfies the constraints.

Problem Progression

The structured interviews with the tutor revealed an almost algorithmic approach to discovering evidence of a student's mastery of relevant skills. The approach centred on asking problems, which were initially selected to be of medium difficulty with respect to what was considered to be an appropriate syllabus. The notion of difficulty was based on the number of identifiable skills required to solve a problem. This strategy is similar to that of Beck *et al.* (1997), though other researchers have developed more complex models (Lee, 1996). The tutor considered a correct answer to support the belief that a student had mastered the use of a skill for problems at a particular level of difficulty. Two pieces of positive evidence or two pieces of negative evidence were required to allow the tutor to reach a firm judgement. Thus at most three question of a specified level of difficulty requiring a particular skill were required. This repetition helps eradicate uncertainty due to the possibility of guesswork or careless slips.

Evidence of the possession of a skill based on performance with problems at a given level of difficulty was assumed to apply to problems of lesser difficulty. Similarly, evidence about the lack of a skill derived from tests with problems at a given level of difficulty was considered to be valid for more difficult problems. The tutor explored a student's knowledge by moving from problems at one level of difficulty to another in the manner of a binary search (that is, by halving the difference of problem difficulties at each step).

The implementation of this strategy is quite straightforward given the problem generating predicates described above, and the end product of the testing can be presented as a cross tabulation of skill competence by problem difficulty.

Conclusion

This work arises from research on the use of teacher emulation to construct an adaptive test for an area in elementary arithmetic. It has discussed the task of knowledge acquisition for the development of adaptive tests, and it has shown how constraint logic programming can be used to support the interaction between a teacher and an interviewer. The central part of this paper describes how this may be achieved by using a particular constraint satisfaction system, clp(FD). This system forms the basis of a software tool used interactively during the acquisition process to capture descriptions of classes of problems and possible answers. From these descriptions the software provides examples of problems during the acquisition process. The examples can be used to clarify or confirm the recorded representations of problems and answers. The semi-automation of the early stages of developing adaptive tests should, by reducing their cost, widen their appeal.

References

Abbott, J. (1995) A Matter of Constraint. Unix News **82** 18-19.

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.), CISM Courses and Lectures no.383. International Centre for Mechanical Sciences. User Modeling. Proceedings of the Sixth International Conference UM97. New York: SpringerWien 277-289.

Carlsson, M., Ottosson, G.,and, Carlson, B. (1997) An Open-Ended Finite Constraint Solver **in** Proc. Programming Languages: Implementations, Logic and Programs.

Collins, J.A., Greer, J.E., and, Huang, S.X. (1996) Adaptive Assessment Using Granularity Hierarchies and Bayesian Nets **in** Frasson, C., Gauthier, G.,and, Lesgold, A. (eds.) Intelligent Tutoring Systems, Third International Conference, ITS'96, Montréal, Canada, June 1996 Proceedings. Lecture Notes in Computer Science 1086. Berlin Heidelberg: Springer-Verlag 569-577.

Dowling, C.E., Hockemeyer, C.,and, Ludwig, A.H. (1996) Adaptive Assessment and Training Using the Neighbourhood of Knowledge States **in** Frasson, C., Gauthier, G.,and, Lesgold, A. (eds.) Intelligent Tutoring Systems, Third International Conference, ITS'96, Montréal, Canada, June

1996 Proceedings. Lecture Notes in Computer Science 1086. Berlin Heidelberg: Springer-Verlag 578-587.

Dowling, C.E. and Kaluscha, R. (1995) Prerequisite Relationships for the Adaptive Assessment of Knowledge **in** Greer, J. (ed.) Proceedings of AI-ED'95, 7th World Conference on Artificial Intelligence in Education, Washington, DC, 16-19 August 1995, AACE 43-50.

Falmagne, J.C., Doignon, J.P., Koppen, M., Villano, M., and, Johannesen, L. (1990) Introduction to Knowledge Spaces - How to Build, Test, and Search Them. Psychological Review **97** (2) 201-224.

Huang, S.X. (1996) A Content-Balanced Adaptive Testing Algorithm for Computer-Based Training Systems **in** Frasson, C., Gauthier, G.,and, Lesgold, A. (eds.) Intelligent Tutoring Systems, Third International Conference, ITS'96, Montréal, Canada, June 1996 Proceedings. Lecture Notes in Computer Science 1086. Berlin Heidelberg: Springer-Verlag 306-314.

Khuwaja, R. (1996) A Model of Tutoring: Based on the Behavior of Effective Human Tutors **in** Frasson, C., Gauthier, G.,and, Lesgold, A. (eds.) Intelligent Tutoring Systems, Third International Conference, ITS'96, Montréal, Canada, June 1996 Proceedings. Lecture Notes in Computer Science 1086. Berlin Heidelberg: Springer-Verlag 130-138.

Lee, F.L. (1996) Electronic Homework: An Intelligent Tutoring System in Mathematics. PhD Thesis. The Chinese University of Hong Kong.

Lightfoot, J.M. (1999) Expert knowledge acquisition and the unwilling expert: a knowledge engineering perspective. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks **16** (3) 141-147.

Linacre, J.M. (1995) Individualized Testing in the Classroom **in** Anderson, L.W. (ed.) International Encyclopedia of Teaching and Teacher Education. Oxford, New York, Tokyo: Elsevier Science 295-299.

Marriott, K. and Stuckey, P. (1998) Programming with Constraints: An Introduction, Cambridge, MA: MIT Press.

McGraw, K.L. and Harbison-Briggs, K. (1989) Knowledge Acquisition, Principles and Guidelines. New Jersey, London: Prentice-Hall, Englewood Cliffs.

Payne, S.J. and Squibb, H.R. (1990) Algebra mal-rules and cognitive accounts of errors. Cognitive Science **14** 445-481.

Rios, A., Millan, E., Trella, M., Perez-de-la-Cruz, and, Conejo, R.(1999) Internet Based Evaluation System **in** Lajoie, S.P. and Vivet, M. (eds.) Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. Volume 50 in Frontiers in Artificial Intelligence. Amsterdam:IOS Press 387-394.

Syang, A. and Dale, N.B. (1993) Computerized adaptive testing in computer science: assessing student programming abilities. Proceedings of the twenty-fourth SIGCSE Technical Symposium on Computer Science Education, February 18-19 1993, Indianapolis USA 53-56.

Wainer, H. (1990) Computerized Adaptive Testing: A Primer. New Jersey: Lawrence Erlbaum Associates.

Wainer, H. and Mislevy, R.J. (1990) Item Response Theory, Item Calibration and Proficiency Estimation **in** Wainer, H. (ed.) Computerized Adaptive Testing: A Primer. New Jersey: Lawrence Erlbaum Associates 65-102.