Constraint-Based Tutors: a Success Story

Antonija Mitrovic
Michael Mayo
Pramuditha Suraweera
Brent Martin

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{tanja,mmayo,psu16,brent}@cosc.canterbury.ac.nz

Abstract. Student modeling (SM) is recognized as one of the central problems in the area of Intelligent Tutoring Systems. Numerous SM approaches have been proposed and used with more or less success. Constraint-based modeling is new approach, which has been used successfully in three tutors developed in our group. The approach is extremely efficient, and it overcomes many problems that other student modelling approaches suffer from. We present the advantages of CBM over other similar approaches, describe three constraint-based tutors and present our future research plans.

Keywords: Intelligent Tutoring, AI application to Design, Knowledge Representation

1 Introduction

It is well known that one-on-one human tutoring is much more effective than traditional classroom instruction, providing an increase of two standard deviations in learning performance [4]. The goal of the research in the area of Intelligent Tutoring Systems (ITS) is to build computer-based tutors that achieve the effects of learning individually with a human tutor. It is a very ambitious goal, and currently available systems are nowhere near achieving it, although there are ITSs that provide an increase of one standard deviation over a longer time period [3].

Student modeling is widely accepted as one of the central problems in ITSs. If a system is to be effective, it must reason about student's knowledge and adapt its actions to the needs and abilities of each individual student. Many SM approaches have been suggested over the years. Some of them are suited to a particular domain, or a teaching strategy. The research presented here focuses on effective and computationally tractable student modeling. We adopted Constraint-Based Modeling, a student modeling approach recently proposed by Ohlsson [15], and have had extremely good experiences with it. We have developed three constraint-based tutors, all supporting problem solving, conceptual and meta-learning. Our focus is on student models; they should be precise enough to guide instruction, and computationally tractable at the same time.

In the next section we present CBM briefly, and then turn to the three implemented systems in the following section. The final section presents conclusions and directions for future work.

2 Constraint-Based Modeling

Student modeling can be defined as the process of gathering relevant information in order to identify and represent the knowledge state of a student. In an ideal case, the model of a student should illustrate his/her knowledge, preferred learning strategies, areas of interest besides that of instruction, preferred presentation style, level of concentration and so on. Several techniques for student modeling have been developed for particular domains, the generality of which is yet to be determined by applying them elsewhere. The survey of SM approaches is outside the scope of such a paper and the interested reader is referred to [7].

The task of building a student model is extremely difficult and laborious, due to huge search spaces involved. Several researchers have pointed to the inherent intractability of the task [8, 15, 16]. Various approaches to dealing with the intractability of student modeling have been introduced. Self [16, 17] recommends such design of the interactions that information necessary for building a student model is provided by the student, and not inferred by the system. Also, it is not useful to be able to identify misconceptions in the student knowledge that cannot be dealt with by the tutor. An ITS should model only what it is capable of using in order to generate remedial or other pedagogical actions.

If the goal is to model student's knowledge completely and precisely, student modeling is bound to be intractable. However, a student model can be useful even if it is not complete and accurate [15, 17, 18]. Even simple and constrained modeling is sufficient for instruction purposes, and this claim is supported by findings that human teachers also use very loose models of their learners, and yet are highly effective in what they do [8, 17]. Anderson [2] also argues against such modeling, saying that it is not very useful for students to be told about the sources of their misconceptions, and that students benefit much more by being given an informative error message.

One of the SM approaches that focus on reducing the complexity of the task is Constraint-Based Modeling (CBM) [15]. CBM is based on Ohlsson's theory of learning from performance errors [14]. CBM focuses on faulty knowledge, realizing that it is not sufficient to describe what the student knows correctly. The basic assumption is that diagnostic information is not hidden in the sequence of student's actions, but in the situation (or the problem state) that the student arrived at. This assumption is supported by the fact that there can be no correct solution of a problem that traverses a problem state, which violates the fundamental ideas, or concepts of the domain. The student model does not represent student's actions, but the effects of his or her actions instead.

Because the space of false knowledge is vast, much more so than the space of correct knowledge, Ohlsson suggest the use of an abstraction mechanism realized in the form of state constraints. A state constraint is an ordered pair (C_r, C_s), where C_r is the relevance condition and C_s is the satisfaction condition. C_r is used to identify the equivalence class, or the class of problem states in which C_r is relevant. C_s identifies the class of relevant states in which C_s is satisfied. Each constraint specifies the property of the domain that is shared by all correct paths. In other words, if C_r is satisfied in a problem state, in order for that problem state to be a correct one, it must also satisfy C_s. Conditions may be any kinds of logical formulas, hence may be constructed from various tests on the problem state in question.

In such a way, CBM represents domain knowledge as a set of state constraints. Constraints define sets of equivalent problem states. An equivalence class triggers the same instructional action; hence the states in an equivalence class are pedagogically equivalent. All correct problem solutions do not violate any of the constraints. A violated constraint signals the error, which translates to incomplete and incorrect knowledge.

CBM does not require a runnable expert module, as many other SM approaches do. This is a very important advantage of CBM, because in many domains it can be very difficult to build the problem solver. CBM-based computerized tutors are able to generate instructional actions even without being able to solve problems on their own, by focusing on violated constraints. Of course, CBM does not prevent an ITS from having a domain module. On the contrary, the existence of a

domain module can be very beneficial to the student, as it can provide the answer to student's questions such as ``What do I do next?''. Expert modules are based on a different type of knowledge (i.e., prescriptive) and hence can be used for such purposes. However, the constraint set, if appropriately represented, might also form the knowledge base of a runnable domain module, because it contains rich information about the domain. We have developed a constraint representation for this purpose, and are currently evaluating its potential [9].

Another advantage of CBM is its computational simplicity. Instead of using complex reasoning as required by other diagnostic approaches, CBM reduces student modeling to pattern matching. Conditions are combinations of patterns, and can therefore be represented in compiled forms, such as RETE networks [6], which are very fast. In the first step all relevance patterns are matched against the problem state. In the second step, the satisfaction components of constraints that matched the problem state in the first step (i.e., relevant constraints) are matched. If a satisfaction pattern matches the state, then the constraint is satisfied, and the ITS is not to take any action. In the opposite case, the constraint is violated. The student model consists of all violated constraints. This technique can be used for both on- and off-line student modeling.

Furthermore, CBM does not require extensive studies of student bugs as in enumerative modeling [1]. Another deficiency of many SM approaches that CBM is not sensitive to is the *radical strategy variability phenomenon*. Namely, some approaches assume that a student systematically uses a single procedure for the task at hand. However, Ohlsson shows [15] that each student applies just one of a family of procedures applicable to the problem at hand, and that the procedure selection strategy is ad hoc. Ohlsson claims that each student knows several procedures at the same time and that he or she can switch between them on different problems. On the other hand, CBM allows students to be inconsistent; it ignores the particular strategy the student applied, since different strategies may result in the same constraint broken. The approach is also neutral with respect to the pedagogy, since different pedagogical actions (immediate or delayed ones) may be generated on the basis of the model.

3 SQL-Tutor

SQL-Tutor is an intelligent educational system, which helps university-level students to learn SQL. The architecture of the stand-alone version of the system is illustrated in Figure 1. For a detailed discussion of the system, see [10, 11, 12, 13]; here we present only some of its features. SQL-Tutor consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller (CBM), which analyzes student answers. The system contains definitions of several databases, and a set of problems and the ideal solutions to them. SQL-Tutor contains no domain module. In order to check the correctness of the student's solution, SQL-Tutor compares it to the correct solution, using domain knowledge represented in the form of more than 500 constraints.

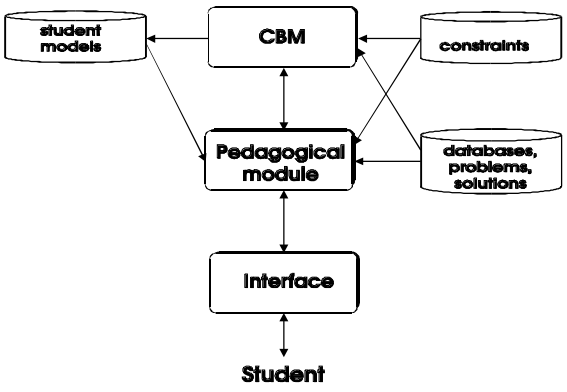

Fig. 1. Architecture of SQL-Tutor

At the beginning of a session, SQL-Tutor selects a problem for the student to work on. When the student submits a solution, the pedagogical module sends it to the student modeller, which analyzes the solution, identifies mistakes (if there are any) and updates the student model appropriately. On the basis of the student model, the pedagogical module generates an appropriate

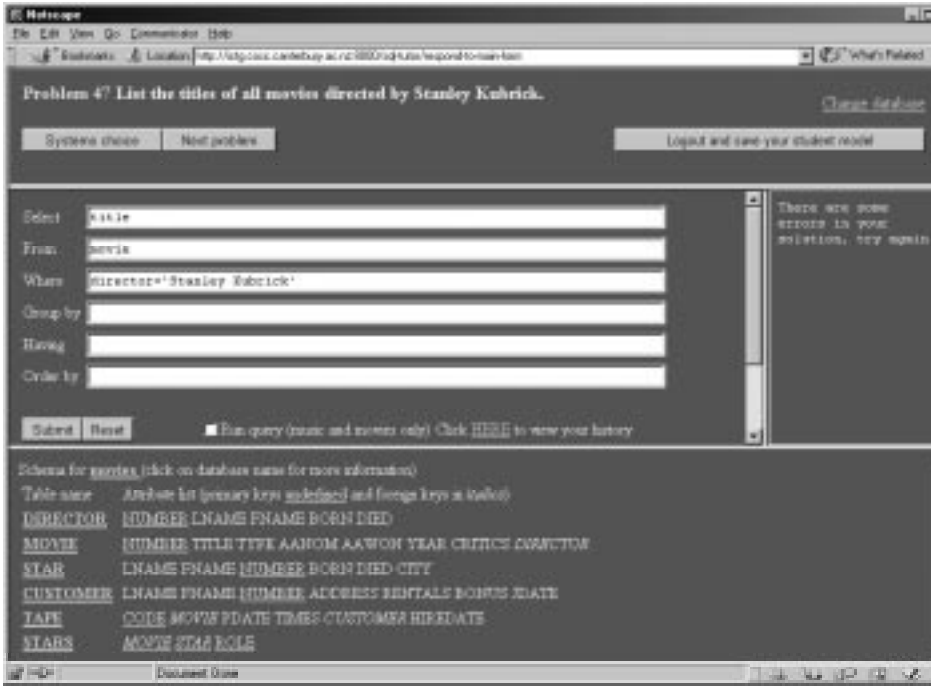

Fig. 2. Interface of the Web-enabled version of SQL-Tutor

pedagogical action (i.e. feedback). When the current problem is solved, or the student requires a new problem to work on, the pedagogical module selects an appropriate problem on the basis of the student model.

The interface of the Web-enabled version of SQLT-Tutor [11], illustrated in Figure 2, has been designed to be robust, flexible, and easy to use and understand. It reduces the memory load by displaying the database schema and the text of a problem, by providing the basic structure of the query, and also by providing explanations of the elements of SQL. The main page is divided into three areas. The upper part displays the text of the problem being solved and students can remind themselves easily of the elements requested in queries. The middle part contains the clauses of the SQL SELECT statement, thus visualizing the goal structure. Students need not remember the exact keywords used and the relative order of clauses. The lowest part displays the schema of the currently chosen database. Schema visualization is very important; all database users are painfully aware of the constant need to remember table and attribute names and the corresponding semantics as well. Students can get the descriptions of databases, tables or attributes, as well as the descriptions of SQL constructs. The motivation here is to remove from the student some of the cognitive load required for checking the low-level syntax, and to enable the student to focus on higher-level, query definition problems. We have also implemented an animated pedagogical agent, which had a very positive motivational effect on the students [13].

Students have several ways of selecting problems in SQL-Tutor. They may work their way through a series of problems for each database, ordered by their complexity, by clicking on the *Next Problem* button. The other option is a system-selected problem (the *System's choice* button), when the system selects an appropriate problem for the student on the basis of his/her student model. We have also implemented a problem-selection strategy based on Bayesian networks [10].

SQL-Tutor has been extensively evaluated. Four evaluation studies have been performed during 1998-2000, which proved that students who learnt with the system achieved significantly higher results than those who did not use the system. An improvement of nearly one standard deviation

was achieved in the 1998 study, which included a single two-hour long session [12]. All the studies showed that CBM has sound psychological foundations and that students acquire constraints at a high rate [12].

4 CAPIT: a punctuation tutor

CAPIT (Capitalisation And Punctuation Intelligent Tutor) is a constraint-based tutor designed for, and evaluated with, school children in the 10-11 year old age group. The system teaches a subset of the basic rules of English capitalisation and punctuation, and currently contains 46 problems and 25 constraints. The problems are relevant to the constraints in roughly equal proportions, although a small number of constraints (such as capitalisation of sentences) are relevant to all the problems. The constraints cover the following parts of the domain: capitalisation of sentences and the names of both people and places, ending sentences with periods, contracting *is* and *not* using apostrophes, denoting ownership using apostrophes, separating clauses using commas, separating list items using commas, denoting direct speech with quotation marks and the correct punctuation of the possessive pronoun *its*.

In CAPIT the student must punctuate and capitalise a fully lowercase, unpunctuated piece of text. If the child makes a mistake, an error message is displayed. For example, Figure 3(a) depicts one of the shorter problems in the ITS. Figure 3(b) shows an incorrect solution submitted by a student, with two errors: the direct speech does not start with a capital letter, and the period is outside the quotation marks. The system displays only one error message at a time, and the student is expected to correct the error and resubmit the problem before any more feedback is displayed. A feedback message would be displayed if the student submitted this solution, e.g. *The full stop should be within the quotation marks! Hint: look at the word* books *in your solution.* Error messages are typically short and relate to only a single mistake, but if the student wants more detailed information, she/he can click *Why?* to be shown further explanatory material.

(a) the teacher said open your books
(b) The teacher said, “open your books”.
(c) The teacher said, “Open your books.”

Fig. 3(a). A problem, (b). a student’s incorrect solution, and (c). the correct solution.

The user interface was designed with the target age group of 10-11 year olds in mind. Two issues of importance when designing interfaces for this age group are facileness (the degree to which the interface is intuitive and self-explanatory), and motivation.

Before starting the exercises, a short tutorial is given to all new users of the system. A child can review the tutorial at any time by clicking the help button. The main interface is then displayed, as depicted in figure 4. Brief instructions relevant to the current problem are clearly displayed at the top of the main interface. This reduces the cognitive load by enabling the child to focus on the current goals without needing to remember them. Immediately below the instructions, and clearly highlighted, is the current problem. In this area, the child interacts with the system by moving the cursor, capitalising letters, and inserting punctuation marks. The child can provide input either by pointing and clicking the mouse, or by pressing intuitive key combinations such as *Shift-M* to capitalise the letter *m*. By requiring the cursor to be positioned at the point where the capital letter or punctuation mark is to go, the child’s ability to locate errors as well as correct them is tested.

Motivation is provided in two forms. Firstly, children accrue points every time they submit a correct solution. The total number of points accrued so far, and the value in points of the current problem, is clearly displayed on the main interface. Secondly, when a correct solution is submitted,

a simple animation is displayed as a reward. We have found this to be adequate motivation for 10-11 year olds.

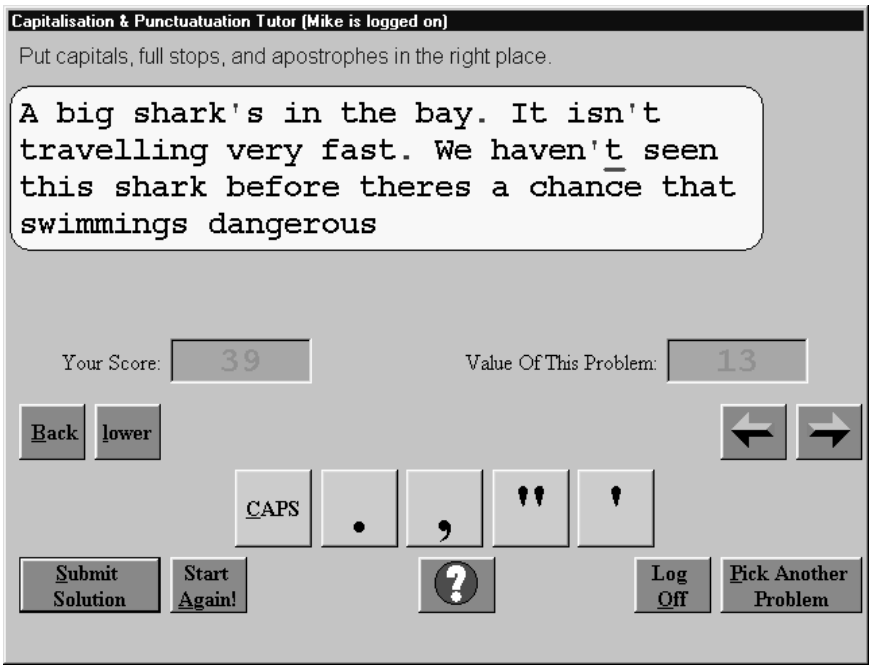

Fig. 4. The tutor's main user interface

The architecture of CAPIT comprises databases of constraints, problems and student models, the user interface, the student modeller, and the pedagogical module. When the student submits a solution, it is passed to the student modeller. The student modeller determines firstly which constraints are relevant to the current solution, and secondly, which constraints are satisfied. The violated constraints are then passed to the pedagogical module so that an error message can be selected.

The pedagogical module solves two key decision tasks: it selects the next problem when the child clicks *Pick Another Problem*, and it selects a single error message for display when the child submits an incorrect solution. A probabilistic student model is used for these decisions. Initially, a student model is set to a population student model, which is later individualized after each student's action. The system uses Bayesian networks and decision theory to decide what problem or feedback to give to the student.

CAPIT was evaluated with three classes of 9-10 year olds at an elementary school in Christchurch, New Zealand, over four-weeks. The first class (Group A) did not use CAPIT. The second class (Group B) used the version of the tutor with randomised problem and error message selection, and the third class (Group C) used the full version of the tutor with the adaptive Bayesian student model. The groups using the tutor, B and C, had one 45-minute session per week. Every student attempt and tutor decision was logged. All groups were administered the same pre- and post-tests. No statistically significant differences were found between the results of the pre-test for the three groups, indicating that the groups were all of comparable ability to start with.

Group	Pre-test (%)	Post-test (%)
A	54.5	47.8
B	58.1	62.7
C	51.0	61.3

Table 1. Mean pre- and post-test scores.

The pre- and post-tests were comparable (and challenging) and consisted of eight exercises similar to those presented by CAPIT, but done with pencil-and-paper. The score for each test was calculated by subtracting the number of punctuation and capitalisation errors from the number of marks for a perfectly correct solution. The mean scores and standard deviations are shown in Table 1. Both Group B and C show an improvement in mean test scores, although the improvement is more marked for Group C. Group A, the class that did not use the tutor, actually regressed. Because the same pair of students in each group completed both a pre- and a post-test, a one-tailed paired difference experiment was performed to gauge the significance of the improvement. Both groups improved significantly (for group B, $\alpha = 0.05$, $t = 1.86$, rejection region ± 1.75; for group C, $\alpha = 0.01$, $t = 3.4$, rejection region ± 2.6). The improvement is thus much more significant for Group C, which used the full version of the tutor.

Figure 5 shows how the students learned the constraints. We looked at the proportion of violated constraints following the *n*th attempt, averaged across all students and all constraints. Just over half of the students were still using the tutor by the 55th attempt, so the data analysis was concluded at this point. Very similar behaviour was observed in SQL-Tutor [12]. Such a smooth learning curve is a proof of psychologically good foundations of CBM.

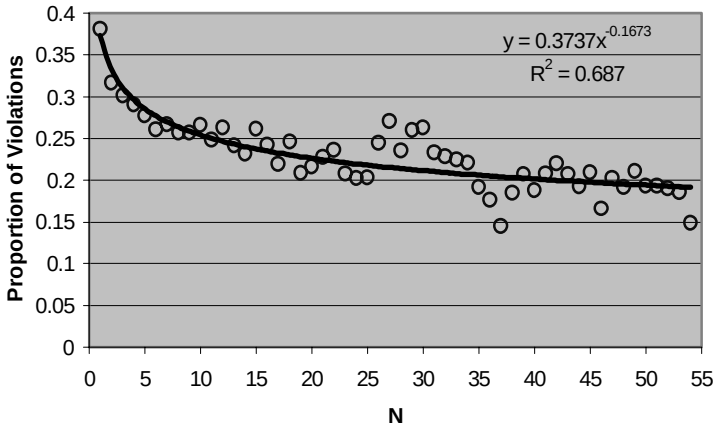

Fig. 5. Mean proportion of constraints that were violated on *n*th attempt.

5 KERMIT: an ITS for database design

Database (DB) modelling is a complicated task, which requires significant amounts of practice to achieve expertise. DB modelling is traditionally taught in a classroom environment where concepts and solutions to typical databases are explained. Our experiences in teaching DB modelling point to the need of providing students with individualized feedback, as students' solutions differ enormously between each other. We have developed KERMIT, an ITS for ER modelling, a popular high-level conceptual data model.

KERMIT is implemented in Microsoft Visual Basic, and supports the entity relationship data model as defined in [5]. The workspace of KERMIT is implemented using Microsoft Visio. KERMIT consists of a user interface, a student modeller and a pedagogical module. The interface of KERMIT is illustrated in Figure 6. It consists of three separate windows that are tiled below each other. The top window is used to display the problem to the student. It also has a drop down list for students to choose their desired level of feedback. The middle window is the workspace for students to model their solution. The feedback window is positioned under the workspace. The feedback window displays textual pedagogical messages whereas the animated agent communicates them verbally. The workspace consists of a stencil developed for ER modelling, which contains all

required constructs. Students can drag and drop constructs on to their workspace to construct their ER model.

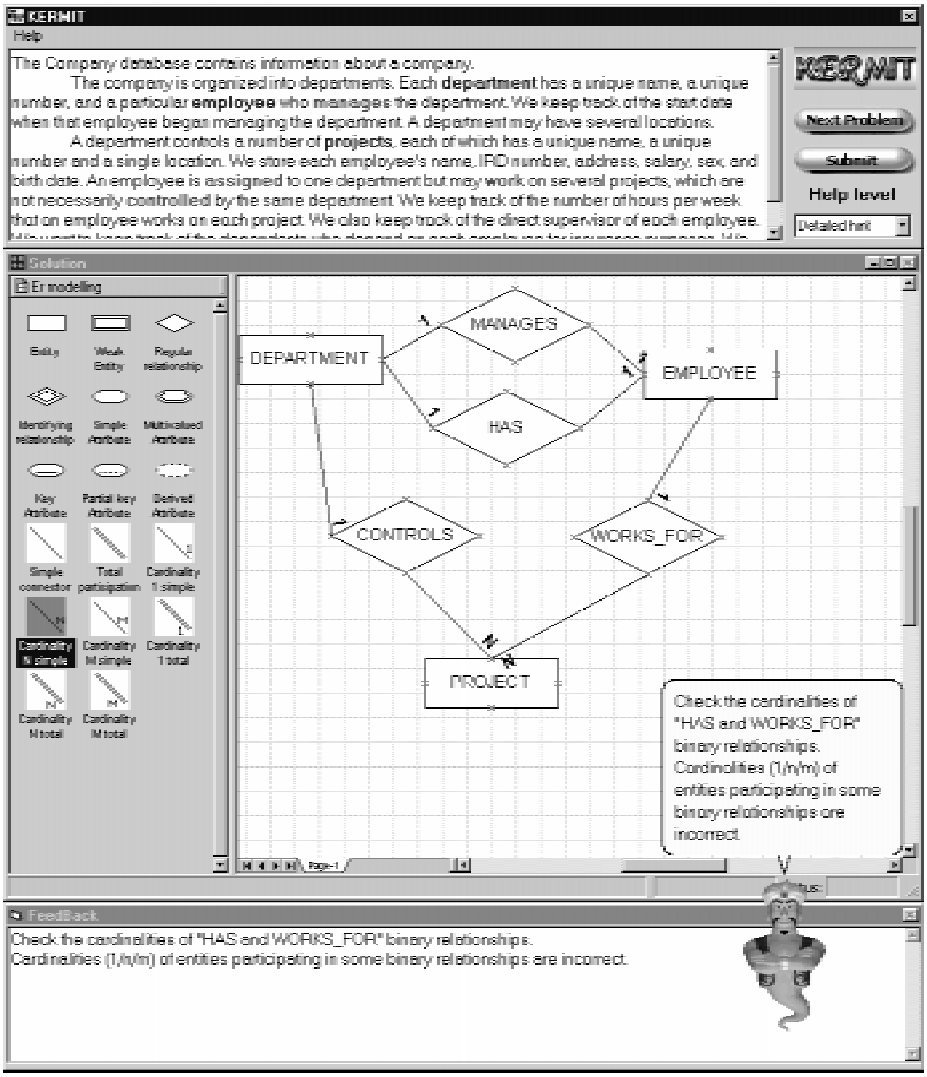

Fig. 6: User interface of KERMIT

When the student has composed their solution to the given problem, the student modeller evaluates it, and after that the PM would generate appropriate feedback depending on the errors. When the current problem is solved the PM selects the next problem that best suits the student model.

The feedback is grouped into six levels in an increasing amount of information: “Correct”, “Error flag”, “Hint”, “Detailed hint”, “All errors” and “Solution”. The first level of feedback simply indicates whether the submitted solution is correct or not. The “Error flag” indicates the type of construct (e.g. entity, relationship, etc.) that is incorrect. “Hint” and “detailed hint” offer instructions on the most important error (chosen from a hierarchy) from all errors. “Hint” is a general message, whereas “detailed hint” provides a more specific message. Feedback on all violated constraints is displayed at the “all errors” level. The complete solution is presented as an image in the final level.

At the current stage, KERMIT's constraint base consists of 90 constraints. These deal with both syntactic and semantic errors. The syntactic constraints vary from simple constraints such as "an entity name should be in upper case", to more complex constraints such as "the weak entity participating in an identifying relationship should have a total participation". The syntactic constraints only concentrate on the student's solution and are independent of the ideal solution. The semantic constraints compare the student's solution to the system's solution. "The student's solution should consist of all the entities present in the ideal solution" is an example of a semantic constraint. The evaluation of KERMIT is planned for early 2001.

6 Conclusions

This paper presented three intelligent tutoring systems that implement Constraint–based Modeling. CBM is a promising new student modeling approach that resolves many of the difficulties present in other diagnostic approaches. It does not require a runnable expert module, which may be difficult or even impossible to develop for some domains. Furthermore, CBM does not require extensive studies of typical errors made by students (i.e. bug libraries) necessary for enumerative bug modeling, and it does not require complex reasoning about possible origins of student errors. CBM is also advantageous over probabilistic methods, such as Bayes networks, which require estimates of prior probabilities. All that CBM requires is a description of the basic principles and concepts in a domain.

From our experiences, CBM is extremely effective and efficient. It may be used in various kinds of domain, such as highly structured procedural tasks and open-ended tasks. CBM does not dictate any pedagogical approach, it equally well supports online and off-line learning, immediate and delayed feedback.

We presented three constraint-based tutors: a system that teaches the SQL database language, a system that teaches punctuation and capitalization rules, and a system for database design. The domains for which these systems are developed are very different, and range from a fairly small set of punctuation rules, which are almost always deterministic, over an artificial language to an open task such as database modeling. Although DB modeling may seem as a much wider domain as the other two, the nature of CBM allows the domain knowledge to be represented as a relatively small set of constraints. In the case of SQL-Tutor, the fact that there are usually several correct ways of solving the problem results in a large knowledge base, containing over 500 constraints. SQL-Tutor and CAPIT were evaluated in real classrooms, and the results show that the students who learnt with the systems achieved significantly higher results on the post-test than those who were in the traditional classroom environment. Furthermore, the analysis of how constraints are learned shows that CBM has a sound psychological foundation.

Current work at ICTG includes enhancements of the presented systems and the development of new constraint-based tutors. We have started developing a generic language tutor, which will be able to support student learning English or a set of foreign languages, when supplied with appropriate constraint sets. Language learning is a task of very different nature to those we have developed tutors for so far, and our primary intention is to evaluate our methodology and CBM in general. We have also started work on an authoring system based on CBM, which will make the development of new tutors much easier.

Acknowledgements

The work presented here was supported by the University of Canterbury research grants U6242 and U6430.

7 References

1. Anderson, J.R., Jeffries, R.: Novice LISP Errors: Undetected Losses of Information from Working Memory. *Human--Computer Interaction*. 22, 403-423, 1985.
2. Anderson, J.R.: *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1993.
3. Anderson, J.R, Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences*, 4(2), 167-207, 1995.
4. Bloom, B.: The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, 3-16, 1984.
5. Elmasri, R., Navathe, S.: *Fundamentals of Database Systems*. Addison Wesley, 1994.
6. Forgy, C.L.: Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19, 17-37, 1982.
7. Greer, J.E., McCalla, G.I. (eds.): *Student Modeling: the Key to Individualized Knowledge--based Instruction*. Berlin: Springer-Verlag, NATO ASI Series, 1994.
8. Holt, P., Dubs, S., Jones, M., Greer, J.E.: 1994, `The State of Student Modelling. In: J.E. Greer and G.I. McCalla (eds.): *Student Modeling: the Key to Individualized Knowledge--based Instruction*. Berlin: Springer-Verlag, NATO ASI Series, 3-35, 1994.
9. Martin, B., Mitrovic, A. *Tailoring Feedback by Correcting Student Answers*. Proc. ITS'2000, G. Gauthier, C. Frasson and K. VanLehn (eds), Springer, pp. 383-392, 2000.
10. Mayo, M., Mitrovic, A.: Using a Probabilistic Student Model to Control Problem Difficulty. In: Gauthier G., Frasson C., and VanLehn K. (eds): *Proc. 5th Int. Conf. ITS'2000*, Springer-Verlag, 524-533, 2000.
11. Mitrovic, A., Hausler, K.: Porting SQL-Tutor to the Web. Proc. ITS'2000 workshop on *Adaptive and Intelligent Web-based Education Systems*, 37-44, 2000.
12. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-based Tutor for a Database Language. *Int. J. on Artificial Intelligence in Education*, 10(3-4), 238-256, 1999.
13. Mitrovic, A., Suraweera, P.: An Animated Pedagogical Agent. In: Gauthier G., Frasson C., and VanLehn K. (eds): *Proc. 5th Int. Conf. ITS'2000*, Springer-Verlag, (2000) 73-82.
14. Ohlsson, S: Learning from Performance Errors. *Psychological Review*. 103 (2), 241—262, 1996.
15. Ohlsson, S.: Constraint-based student modeling. In: Greer, J.E., McCalla, G (eds): *Student modeling: the key to individualized knowledge-based instruction*, 167-189, 1994.
16. Self, J. A.: Bypassing the intractable problem of student modeling. In: C. Frasson and G. Gauthier (eds.*), Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*. Norwood: Ablex, 107-123, 1990.
17. Self, J.: Formal Approaches to Student Modeling'. In: J.E. Greer and G.I. McCalla (eds.): *Student Modeling: the Key to Individualized Knowledge--based Instruction*. Berlin: Springer--Verlag, NATO ASI Series, 295-352, 1994.
18. Woolf, B.P., Murray, T.: Using Machine Learning to Advise a Student Model. In: J.E. Greer and G.I. McCalla (eds.): *Student Modeling: the Key to Individualized Knowledge--based Instruction*. Berlin: Springer--Verlag, NATO ASI Series, 127-146, 1994.