Authoring Web-Based Tutoring Systems with WETAS

Brent Martin and Antonija Mitrovic

ICTG, Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{bim20,tanja}@cosc.canterbury.ac.nz

Abstract. Constraint-Based Modelling (CBM) is a student modelling technique for Intelligent Tutoring Systems (ITS) that is especially suited to complex, open-ended domains. It is easier to build tutors in such domains using CBM than other common approaches. We present WETAS (Web-Enabled Tutor Authoring System), a tutoring engine that facilitates the rapid implementation of ITS in new domains using CBM. We describe the architecture of WETAS and give examples of two domains we have implemented. We also present the results of an evaluation of a tutoring system built using WETAS in a New Zealand school.

1 Introduction

Constraint-Based Modelling (CBM) [8] is an effective approach for building Intelligent Tutoring Systems that simplifies the building of domain and student models. We have used CBM to develop SQL-Tutor [5], an ITS for teaching the SQL database language. SQL-Tutor tailors instructional sessions in three ways: by presenting feedback when students submit their answers, by controlling problem difficulty, and by providing scaffolding information. Students have shown significant gains in learning after as little as two hours of exposure to this system [6].

While CBM reduces the effort of building domain models for ITS, the task of building an ITS is nevertheless still large. To reduce the authoring effort we have developed WETAS (Web-Enabled Tutor Authoring System), a web-based tutoring engine that performs all of the common functions of text-based tutors. To demonstrate the flexibility of WETAS we have re-implemented SQL-Tutor and developed a new Language Builder ITS (LBITS). Although these domains share the property of being text-based, they have very different problem/solution structures. We have evaluated LBITS in a New Zealand school.

2 ITS and Constraint-Based Modelling

Intelligent Tutoring Systems are educational systems that provide one-on-one tutoring by tailoring their behaviour to individual student needs, or “caring” about their students [10]. There are many different approaches, but most share the general characteristics of domain and student modelling. The domain model encapsulates the principles of the subject being taught. This information is used to provide detailed feedback about student misconceptions and to drive the pedagogical process. The student model relates each student to the domain. For example, overlay models indicate which parts of the domain model the student has mastered. We are particularly interested in ITS that foster “learning by doing”. These systems give students problems to work on and provide individual help while they try to solve them.

CBM [8] is a relatively new approach to domain and student modelling, based on the theory of learning from performance errors [9]. It models the domain as a set of state constraints, where each constraint represents a declarative concept that must be learned and internalised before the student can achieve mastery of the domain. Constraints represent restrictions on solution states, and take the form:

\[
\text{If } \text{<relevance condition> is true for the student's solution,} \\
\text{THEN } \text{<satisfaction condition> must also be true}
\]

The relevance condition of each constraint is used to test whether the student’s solution is in a pedagogically significant state. If so, the satisfaction condition is checked. If it succeeds, no action is taken; if it fails, the student has made a mistake, and appropriate feedback is given. Ohlsson does not impose any restrictions upon how constraints are encoded and/or implemented. We have used a pattern-matching representation designed for this purpose [3]. For example, the following constraint from the SQL domain checks that the names used in the “WHERE” clause are valid names from the database:
"You have used some names in the WHERE clause that are not from this database."

(match SS WHERE (?* (^name ?n) ?*))         ; relevance condition
(or (test SS (^valid-table (?n ?t)))       ; satisfaction
 (test SS (^attribute-p (?n ?a ?t))))      ; condition
"WHERE")

The constraint language is simple to use and allows the system to reason about the constraints in three ways: it may evaluate the student solution against them to decide what is wrong and give feedback, it may use the constraints to correct errors in the student’s input (and thus show them how to proceed), and it may use them to generate new problems to present to the student. For more information, see [3, 4].

3 Architecture

WETAS is a web-based tutoring engine that provides all of the domain-independent functions for text-based ITS. It is implemented as a web server, written in Allegro Common Lisp, and using the AllegroServe Web server (see www.franz.com). WETAS supports students learning multiple subjects at the same time; there is no limit to the number of domains it may accommodate. Students interact through a standard web browser such as Netscape or Internet Explorer. Figure 1 shows a screen from SQL-Tutor implemented in WETAS. The interface has four main components: the problem selection window (top), which presents the text of the problem, the solution window (middle), which accepts the student input, the scaffolding window (bottom), which provides general help about the domain, and the feedback window (right), which presents system feedback in response to the student’s input.

WETAS performs as much of the implementation as possible, in a generic fashion. In particular, it provides the following functions: problem selection, answer evaluation, student modelling, feedback, and the user interface. The author need only provide the domain-dependent components, namely the structure of the domain (e.g. any curriculum subsets), the domain model (in the form of constraints), the problem/solution set, the scaffolding information (if any), and possibly an input parser, if any specific pre-processing of the input is required.

WETAS provides both the infrastructure (e.g. student interface) and the “intelligent” parts of the ITS, namely the pedagogical module and the domain model interpreter. The former makes decisions based on the student model regarding what problem to present to the student next and what feedback they should be given. The latter evaluates the student’s answers by comparing them to the domain model, and uses this information to update the student model.

Problems are selected by determining the difficulty of each candidate exercise with respect to the student. When the problems are authored, the system compares them to the domain model to determine their structural complexity, which is based on the number and complexity of the constraints that are relevant to them. When a new problem is to be selected, each candidate is compared to the student model, and the difficulty is increased by a constant amount for each constraint the student currently doesn’t know, and by a different constant for each constraint that has never been relevant. Thus, the difficulty of each problem differs according to each student’s strengths and weaknesses. The system then chooses the one with a difficulty that is closest to the student’s current ability rating.

4 Example Domains

WETAS has been implemented in prototype form and used to build two tutors, to explore its capabilities and evaluate its effectiveness in reducing the ITS building effort. These two implementations are now described.

SQL-Tutor. SQL-Tutor [5] teaches the SQL database query language to second and third year students at the University of Canterbury, using Constraint-Based Modelling. Students are given a textual representation of a database query that they must perform, and a set of input fields (one per SQL clause) where they must write an appropriate query. This system was implemented in 1998 as a standalone tutor, in 1999 as a Web-enabled tutor, and has been re-implemented in WETAS. The general design and interface of WETAS borrows heavily from the original SQL-Tutor. We had no problems implementing SQL-Tutor in WETAS.

Language Builder ITS (LBITS). Language Builder is an existing paper-based teaching aid that is currently being converted to a computer system. It teaches basic English language skills to elementary and secondary school students by presenting them with a series of “puzzles”, such as crosswords, synonyms, rhyming words and plurals. We created an ITS from Language Builder (LBITS) by adding a domain model so that feedback could be expanded from a simple right/wrong answer to more detailed information about what is wrong, such as that the meaning of their answer didn’t match the meaning of the clue, or they have got the letters “i” and “e”
reversed. These rules were inferred from a school spelling resource book [2]. We generated the problem set automatically from a vocabulary list in [2]. The constraints were authored in just five hours, and the problem set was created in around two days. LBITS contains 315 constraints and 222 puzzles. Figure 2 shows LBITS in action.

We have so far implemented a subset of the Language Builder puzzles, all of which share the common format of a set of clues, each requiring one or more single-word answers. The system has been evaluated at a New Zealand elementary school. We now present the results.

5 Evaluation of LBITS

We evaluated LBITS to determine whether the WETAS authoring system facilitates the rapid generation of effective learning systems. We tested the system on nine children aged 11 and 12 from Akaroa Area School. This evaluation was formative only: we were only interested in whether or the students appeared to be learning
from the constraints, and in what they thought of the system. We measure the former by plotting the error rate of individual constraints. If the constraints represent concepts being learned, we expect to see a “power law” [1, 7]. The better the fit to a power curve, the more likely that learning took place, and that our model represents the underlying principles of the domain being taught.

We evaluated the LBITS tutor in March 2002. The students were randomly chosen from an Akaroa School year 11 class. We presented the students with a five-minute introduction to the system, after which they were free to log in and begin using the tutor. We provided anonymous usernames so the students would not feel that the results might be scrutinised by their teachers. The children then used the system for approximately fifty minutes, after which they were instructed to log off. In the last five minutes they were given a questionnaire to fill in, which contained the following questions, where they ticked the appropriate boxes:

- What puzzle(s) did you play? (Scrambled Words / Last Two Letters)
- How were the questions? (too hard / about right / too easy)
- How easy was the software to use? (easy to use / OK / hard to use)
- Did you enjoy using LBITS? (yes, it was fun / it was OK / no)
- How much do you think you learned? (a lot / a little / nothing)

All nine students used the system for the entire session, and answered the questionnaires in full. Note that some students ticked more than one box, if they felt that a single answer was not representative. Table 1 summarises the survey results.

<table>
<thead>
<tr>
<th>Which Puzzle</th>
<th>Difficulty</th>
<th>Ease of use</th>
<th>Enjoyable?</th>
<th>Learned?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrambled: 9</td>
<td>Too easy: 2</td>
<td>Easy: 9</td>
<td>Fun: 8</td>
<td>A lot: 7</td>
</tr>
<tr>
<td>Last two: 2</td>
<td>About right: 8</td>
<td>Okay: 0</td>
<td>OK: 1</td>
<td>A little: 2</td>
</tr>
<tr>
<td></td>
<td>Too hard: 2</td>
<td>Hard: 0</td>
<td>No: 0</td>
<td>None: 0</td>
</tr>
</tbody>
</table>

Table 1. LBITS Survey Summary.

These results show that the system was positively received. All nine students found the system easy to use, and eight thought it was fun. Seven of the nine students thought they learned a lot. The difficulty of the problems set appears to have been about right, with seven students ticking the “About right” box, one ticking both “too easy” and “too hard”, and the other selecting all three choices. These latter students also wrote comments indicating that they thought the problems were mostly right, but sometimes too easy or too hard.

The objective analysis suggested that the students had learned the constraints. Figure 3 shows the “power
curve” obtained when plotting the average constraint error rate versus the number of times this constraint has been relevant. This curve is obtained by observing, for all constraints/students, what proportion of the time a constraint is violated the first time it is relevant to a problem. The same is repeated for the next time, and so on. Note that the power curve tends to deteriorate as the number of relevant problems grows, for two reasons. First, as the number of problems increases, the number of constraints that were relevant for this many problems decreases exponentially, until there are only a few left. At this point a single constraint failure represents a large proportion of the number of constraints still being considered. Second, students sometimes violate constraints even when they have learned them (slips), and occasionally guess the correct response. The error rate therefore trends towards this “random performance” rate over time, rather than ideally decaying to zero. We therefore plotted only the initial six problems for each constraint, which is one less than the average number of problems solved during the session, so is still realistic. The curve obtained exhibits a very good fit to a power curve ($R^2 = 0.833$), and indicates that learning is taking place: students are 30% likely to violate a constraint that they have never used before, whereas after being exposed to a constraint once, this probability drops to less than 15%.

6 Conclusions and further work

Constraint-Based Modelling (CBM) is an effective approach that simplifies the building of domain and student models for some domains, particularly open-ended ones. We have developed a prototype authoring system called WETAS for CBM tutors, which we intend to use to develop additional tutors for further research into CBM and for release into classrooms. We have reimplemented SQL-Tutor using WETAS and built a new tutor, LBITS.

The LBITS tutor was well received by a New Zealand class. Despite being authored in a very short time, it was demonstrated to be effective in that students appeared to be learning the constraints during the evaluation, which suggests it would have a positive effect on their performance in English spelling and vocabulary. Further, the students had a lot of fun and were very motivated, which is a positive outcome in itself.

The WETAS system is still a prototype. We are now developing it into a useable tool, both for our own research and for education providers. We plan this year to use it in a Canterbury University graduate programme teaching ITS: the students will use WETAS to build novel tutors and evaluate their results. We aim eventually to put WETAS online, such that other researchers and educators may deploy their own tutors on our site. The enthusiastic reception given to LBITS at Akaroa Area School also prompts us to explore other domains that we could build useful tutors for at the primary and secondary school levels.

WETAS draws upon the strengths of CBM, plus research carried out to date in practical implementations of CBM. It appears to be a promising tool for the rapid development of new tutors, and a significant step towards the large-scale deployment of Intelligent Tutoring Systems.

Acknowledgement
This research is supported by the University of Canterbury, grant U6430.

References