Domain Modelling with Ontology: A Case Study

Brent Martin, Antonija Mitrovic and Pramuditha Suraweera

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

Abstract. Authoring ITS domain models is a difficult task requiring many skills. We explored whether modeling Ontology reduces the problem, by giving the students of an e-learning summer school the task of developing the model for a simple domain in under sixty minutes using ontology. Some students also used our tool to develop a complete tutor in around eight hours, which is much faster than they could be expected to author the system without the tool. The results suggest this style of authoring can lead to very rapid ITS development.
Keywords: intelligent tutoring systems, authoring systems, constraint-based modeling, domain models, Ontology

1 Introduction

Intelligent Tutoring Systems increasingly show promise as a technology that will expand the horizons of education from those able to attend a bricks-and-mortar institution to anyone with an Internet connection. Acting as an enhancement to traditional distance learning offerings, they promise to augment laboratories and tutorials by allowing students to practice the skills they are learning from home. In recent years tutors such as the Geometry and Algebra tutors, and the Addison-Wesley database place suite (SQL-Tutor, ER-Tutor and NORMIT) have made it out of the lab and into the classroom [1], [2].

Despite this success, intelligent tutors have still not been adopted widely. One reason for this is the difficulty in building them. Recent research efforts have tried to address this shortcoming. The Cognitive Tutor Authoring Tools (CTAT) [3] attempt to reduce the authoring effort for ITSs based on model tracing. The tools support the creation of two types of tutor: 'Pseudo tutors' and 'Cognitive tutors'. Authors can quickly build Pseudo tutors by developing the user interface for the tutor and then demonstrating the solution to one or more problems. However, these are not "real" ITSs: the resulting model is suitable only for the problems from which it was authored: essentially they are just traces of possible behaviour for that problem. To convert them into full cognitive tutors, the author must manually create production rules that represent a general model of the domain. REDEEM [4] allows educators to add pedagogy to e-learning delivery by tailoring the delivery of educational material to stereotypical student groups. Diagnostic models are not supported however.

Constraint-Based Modelling (CBM) [5] is an effective approach for building Intelligent Tutoring Systems (ITS) that supports the building of domain and student models. Constraint-based tutors are effective: students using SQL-Tutor have shown significant gains in learning after as little as two hours of exposure to this system [6]. Also, CBM seeks to minimize the authoring effort by requiring the author to model only states, rather than solution paths [7]. Nevertheless, the task of building an ITS is still large. To reduce the authoring effort we 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 re-implemented SQL-Tutor and developed a new ITS for teaching spelling and vocabulary (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 and found it to be effective [8, 9].

WETAS removes much of the effort required to build an ITS, but it does not directly facilitate the building of the domain model, which is arguably one of the most difficult tasks [10]. In

particular, the author must write the domain rules or "constraints", which requires a level of programming skill. For complex domains the constraint set can quickly become large (SQL-Tutor has over 600 constraints), making it hard to manage. One way to overcome this is by modeling the domain at a higher level using ontology. We developed a tool, WETAS-Ontology, which allows authors to graphically model the domain as ontology. A constraint generator then creates the required constraints from the concepts in the ontology. The resulting constraints form a domain model that can be used to provide highly specific feedback that is tailored to the individual student's misconceptions, and to drive the pedagogical process, for example by selecting problems based on the concepts for which the student is currently violating constraints. The ontology assists in this latter task by allowing the problem selector to infer similar concepts a student is likely to find difficult, when the problems for the current concept have been exhausted.

WETAS-Ontology was used as a learning aid at the 2006 e-learning school at the National College of Ireland, which enabled us to test our hypothesis that modeling in ontology is easier and faster than writing constraints by hand. This paper reports on our experiences. The next section briefly introduces CBM and the WETAS authoring shell, and WETAS-Ontology is described in Section 3. Sections 4 and 5 describe how WETAS-Ontology was used at the e-learning summer school and discuss what we learned from this experience. We conclude in Section 6 and discuss limitations of the tool.

2 Constraint-Based Modelling and WETAS

CBM [5] is based on the theory of learning from performance errors [8, 9, 11]. It models the domain as a set of state constraints, where each constraint represents a declarative concept that must be learned and internalized before the student can achieve mastery of the domain. Constraints represent restrictions on solution *states*, and take the form:

If <*relevance condition*> *is true for the student's solution,*
THEN <*satisfaction condition*> *must also be true*

The relevance condition of each constraint checks 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. *Syntactic* constraints check that the solution is syntactically correct. Conversely, *semantic* constraints check whether the student's solution has solved the problem, usually by comparing it to the "ideal" solution supplied by the teacher. The constraints implicitly encode semantics by testing for all of the different possible encodings of the semantic concept they are attempting to test. The student is thus permitted to use a different problem-solving strategy to the author, or even to mix strategies, provided no fundamental domain concepts are violated.

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. 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 modeling, 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. Constraints are written in a custom pattern-matching language that is intended to be simple to author. The system reasons about the constraints in three ways: it may evaluate the student solution against constraints 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 constraints to generate new problems to present to the student. For more information, see [12, 13].

WETAS has been used to build several tutors, including EER-Tutor for Addison-Wesley [2] and Collect-UML [14]. It has also been used for the past four years by a graduate University class in Intelligent Tutoring Systems at Canterbury University. In this class, students are assigned the task of building an ITS in WETAS. The first time it was used by this class it became apparent that further authoring tools are required: the students were able to build a tutor in the time allocated (three weeks) but their domain models were generally sub-optimal [15]. We found that students make mistakes at all levels of the domain authoring process: they fail to model pedagogically significant states, fail to capture the intended states in their constraints, and make errors during constraint encoding or encode them inefficiently. Our proposed solution is to provide a high-level tool that automates the encoding of constraints based on an ontology that the author provides. We hypothesize that this will help in two ways: by removing the low-level steps from the authoring process (and thus preventing encoding errors being made) and by allowing the author to visualize the domain so that they are more likely to capture the intended pedagogical states.

3 WETAS-Ontology

The use of ontology in education systems is not new. Mizoguchi and Bourdeau [16] advocate authoring intelligent instructional systems by engaging authors in knowledge *modeling* rather than knowledge engineering. They propose building education systems by creating *task* ontology (which models pedagogy) and *domain* ontology, which represents each individual domain. We are interested in the latter: how do we use ontology to develop domain/student models?

WETAS-Ontology is an experimental tool for authoring ITS domains. It consists of two parts – a graphical editor for creating the ontology, and a constraint generator. The latter parses the ontology and creates constraints for testing the student solution based on the concepts in the ontology. One goal of this research is to develop authoring tools that are easy for "laymen" to use. Many tools already exist that facilitate the development of ontology (e.g. Protégé [17], Protégé-Owl [18]), however these tools are typically aimed at experienced knowledge engineers, and we considered they would be too difficult for non-experts to use. In particular, our tool attempts to graphically visualise the entire model in a clear, graphical manner.

Fig. 1 is a screen shot of the ontology editor showing the developed ontology for the domain of search engine queries, which was used for the case study. In this domain students are given the criteria for a search engine query, which they write using a hypothetical language that consists of logical expressions containing the words and strings they are looking for. The ontology is a combination of taxonomy (kind-of relationships) and partonomy (part-of relationships). The graphical representation adopted was chosen to visualize both of these views simultaneously. Diamonds represent alternative constructs/concepts (kind-of relationships). For example, a *search expression* consists of a *negative expression* or a *positive expression*. Conversely, child nodes of rectangles represent a strict sequence of required sub-parts. In this ontology, a *negative expression* consists of NOT, followed by a *left bracket*, followed by a *positive expression* followed by a *right bracket*. Rectangles with double-lined sides represent concepts that have already been defined elsewhere in the ontology; domains may thus be recursive, as in the example given. A concept may optionally have three properties: *role*, which identifies their purpose in the parent concept (for example, the role of search expression in a complex expression could be "first argument"); *reversible*, which indicates whether or not this concept has the same meaning when parsed backwards; *pattern*, which describes how this concept is identified in the solution. Pattern may be a string, or the name of a macro if complex processing is required to determine membership of the concept. The purpose of the ontology is to capture the fundamental concepts of the domain so that these can be tested in the student solution. There is no standard process for creating ontology, however we have found that for many domains (including the one in Fig. 1) a useful approach is to begin with the grammar of valid solutions and add further concepts as required.

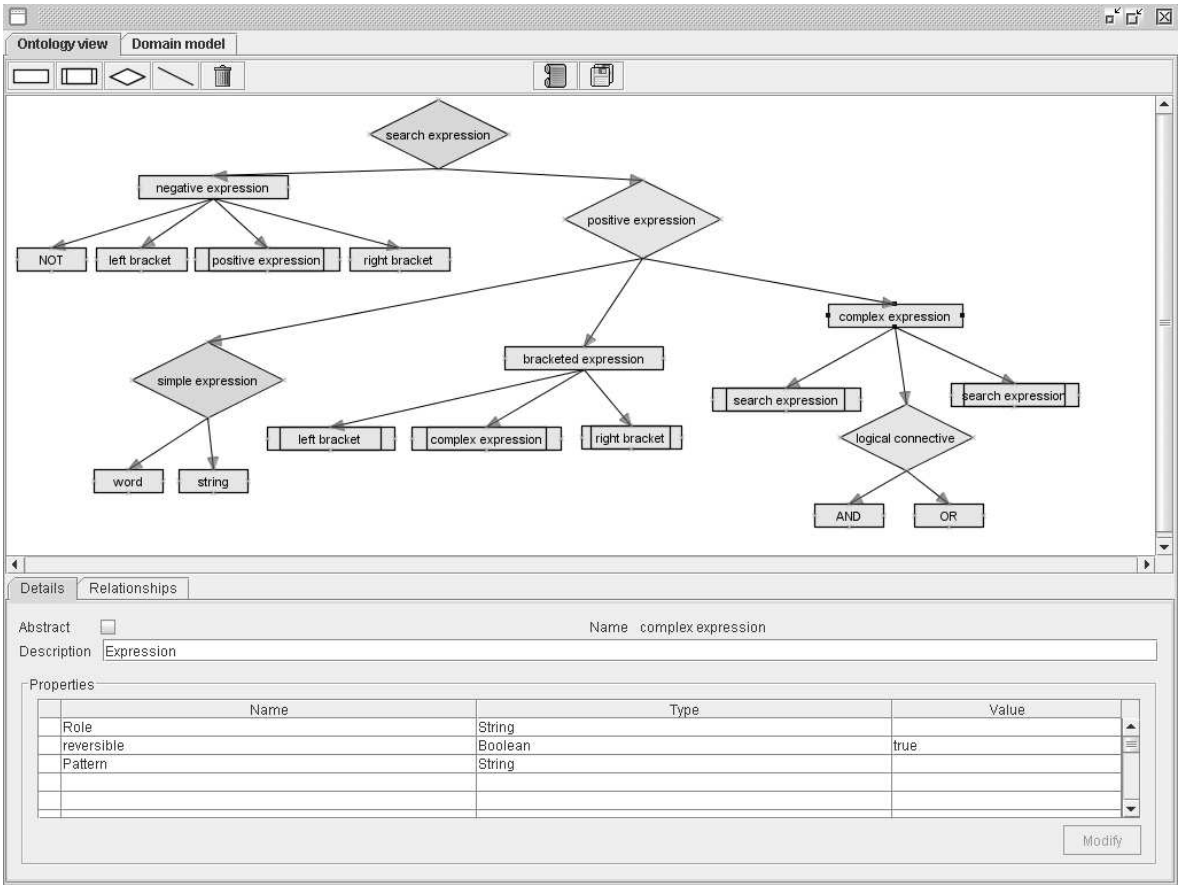

Fig. 1. WETAS-Ontology interface

The constraint generator uses the ontology to create a set of constraints that can be loaded into WETAS and used to evaluate student solutions. For the purpose of this study we generated only semantic constraints. Templates are used to create a set of constraints from each concept in the ontology. These constraints test for the presence/absence of any examples of each concept (i.e. is this concept used at all), that all of the required instances of each concept are present in the solution, and that the subcomponents of each instance are correct (e.g. does the student's logical connective have the correct arguments). Feedback messages are also generated automatically based on templates. This fairly simple set of templates yields a plausible domain model. Note however that it is not intended to deliver the final set of constraints; typically authors will modify the feedback messages, add additional constraints for complex concepts and edit the generated constraints, perhaps to make them more general. Fig. 2 shows two examples of generated constraints. The first checks whether or not a string is needed. The test for a string is complex, so a macro has been used. Writing the macro is an additional task to producing the ontology; in practice few (if any) macros are required. The second constraint checks whether or not a complex expression is needed. In this case there is no easy way to test for this concept because it consists only of two alternative sub-concepts and no literal components. The generator therefore descends the tree until it finds sub-concepts with literal components (in this case "AND" and "OR") and creates alternative tests for each alternative sub-component.

WETAS-Ontology was evaluated at the e-learning summer school in June 2006 at the National College of Ireland, Dublin. This forum was considered an ideal testing ground because the participating students were of mixed backgrounds, with less than half being Computer Scientists. The first author gave instruction at this school, which consisted of two hours of lectures about ITS (and domain/student modeling in particular), followed by a 90 minute practical session. Instructors were also asked to contribute a potential project idea, from which the students would choose one for a one-day practical project. WETAS-Ontology was used for both of these purposes.

```
(5 "Check whether you need one or more string(s) in your answer."
  (MATCH IS ANSWER (?* (^string ?IS_1) ?*))
  (MATCH SS ANSWER (?* (^string ?SS_1) ?*))
"ANSWER")

(16 "Are you sure you need complex expression(s) in your answer?"
 (OR (MATCH SS ANSWER (?* "AND" ?*)) (MATCH SS ANSWER (?* "OR" ?*)))

 (OR (MATCH IS ANSWER (?* "AND" ?*)) (MATCH IS ANSWER (?* "OR" ?*)))
"ANSWER")
```

Fig. 2. Generated constraint examples

4 Case Study 1: A Domain Model in Sixty Minutes

To determine the feasibility of using WETAS-Ontology we asked the students to use it during their 90-minute practical session to develop the ontology for the domain of search engine queries. The students were first lectured about the tool, WETAS and the domain; this took approximately 30 minutes. They then had a further 60 minutes to develop their model. The WETAS tutor authoring shell was installed on each of their computers along with WETAS-Ontology. The other necessary components of the search engine query tutor (e.g. the problem/solution set) were also set up for them. The students could therefore test their ontology at any time. First, they would use the ontology editor to begin creating the ontology. They then instructed the tool to generate the constraint set. Finally, they loaded the constraints into WETAS and tried out the tutor. They were able to repeat this procedure as often as desired until they had completed the model or ran out of time. When they first used WETAS-Ontology it contained just the definition of "simple expression" from Fig. 1. Twelve students attempted to complete the domain model. We categorized each model by comparing to that in Fig. 1. The categories were: *complete* – the model leads to as good a constraint set as ours; *useable* – the model generated a significant subset of the constraints, such that the resulting tutor gave useful feedback; *good attempt* – the model had a significant number of the relevant concepts but contained substantial errors or omissions; *poor* – some attempt had been made but there had been little progress.

Half of the students produced useable domain models, of which one was almost identical to that produced by this paper's authors, and was both of high quality and complete. Fig. 3 shows a useable (but not complete) model. The main problem is that the student has not made the model recursive; the resulting constraints are therefore unable to deal with complex solutions. For example, because the arguments to "disjunct" are simple expressions only, the model generated from this ontology will be able to recognize "fish and chips" but not "fish and chips and salt". There were also other differences, such as whether or not the author had grouped conjuncts and disjuncts into a high level concept (e.g. "complex expression" in Fig. 1). Some of the "useable"

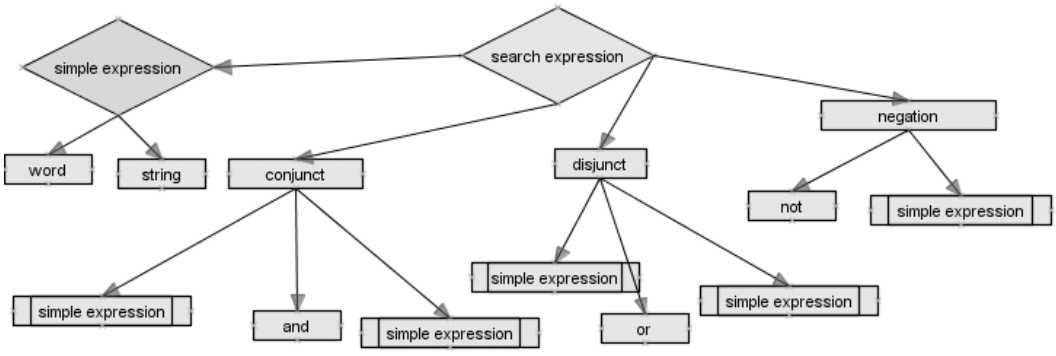

Fig. 3. Example of a "useable" ontology

solutions also missed whole parts of the ontology, such as bracketed expressions, or duplicated parts of the ontology instead of abstracting out common concepts. Of the other six participants, three were classified as “good attempt”. These students had produced reasonable ontologies but they were still some way from being complete, and would hence generate constraint sets that failed to test significant features of the solution. The remaining three were “poor”, and appeared to have struggled with the whole task of creating ontology.

When asked informally for their comments the students were generally very positive about the experience. In particular, they were impressed that they had produced an ITS that generated useful feedback in such a short space of time (less than 60 minutes). They also commented that they found the tool easy to use and that the ontology representation, once explained, was easy to understand. However, it appears that this approach to authoring does not suit everyone. In particular, concepts such as recursion appeared not to be obvious to most participants. For those participants who scored “poor” it is likely that they have not had to perform similar abstraction tasks before. This may be a feature of the students’ background; the student who developed a complete ontology was a Computer Scientist studying in a similar area. For authors of other background some tuition in developing ontology will likely be needed.

Despite these limitations the results were sufficiently positive that we proposed WETAS-Ontology as a potential project.

5 Case Study 2: A Tutor in a Day

The participants at the summer school were all required to contribute to a group project, which would be assessed. The students were given a list of seven potential projects spanning a variety of subjects in the general area of e-learning. Eight of the students (more than half the class) chose to use WETAS-Ontology to build an ITS. They separated into two groups, both of which worked on tutors in the domain of English spelling, a similar domain to an example they had been given. The goal was to build a complete tutor from scratch. They were allocated around six hours of class time to complete the project, although they could work outside class hours if they wished. The first group critiqued WETAS-Ontology and suggested an alternative approach, of which they built a simple prototype. The second group developed a complete tutor using WETAS-Ontology. We turn our attention to this second group.

To develop a complete tutor, the students were required to develop a set of problems and ideal solutions (used by the constraints to check semantics), as well as develop a complete domain model. For this latter task they would need to edit the generated constraints to provide better feedback and to add any additional semantics that were too difficult (if not impossible) to model in the ontology. The group chose the domain of pluralization of nouns. Fig. 4 shows their completed ontology. This ontology is generally similar to what we would have produced, the main difference being that the final leaf nodes are not actually required. There were also some other minor errors (e.g. bacillus and cactus are two examples of the same rule). The leaf nodes on the left represent regular nouns that can be grouped into “rules” of pluralization, while those on the right are irregular nouns that can only be learned individually. The semantics for the regular noun groups

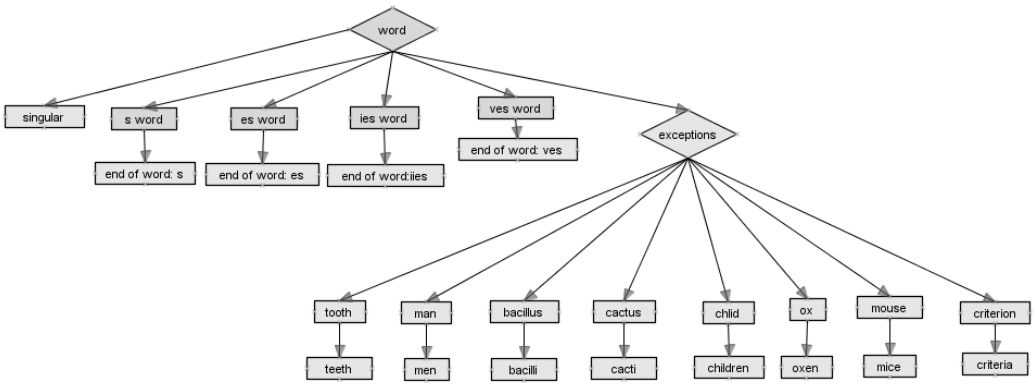

Fig. 4. The pluralization ontology

can either be modeled in the ontology (e.g. by enumerating all of the words belonging to each group) or via macros. The group chose to use this latter solution, and two of the group members paired up to perform this job. There are two ways that macros can be used to represent the required concepts: by testing the letters on the end of the word for the required regular form, or by enumerating the words that belong to each group. The former is more robust and efficient but requires a greater knowledge of WETAS' pattern-matching language, while the latter is brittle; the macro will require modification every time new vocabulary is introduced into the problem set. The students chose to enumerate the members of each group, which is understandable given the limited time available, and the fact that they were not taught how to write complex macros.

The group produced a fully working tutor in around eight hours. Whilst the domain chosen was not particularly complex, this is nevertheless an impressive achievement. The final tutor consisted of 108 constraints. If all of the group's time was spent purely on this task, this result equates to less than five minutes per constraint. Given that there were four group members, this equates to around twenty person-minutes. This is significantly less than the 10 hours per rule reported for model tracing tutors [19] or even the 1.1 hours that it may typically take to hand-write a constraint for CBM [7]. For simple domains such as this, authoring by ontology delivers a major improvement in efficiency. The quality of the domain model they produced is comparable to what this paper's authors would have created.

6 Conclusions

ITS authoring is a difficult task. The WETAS tutoring shell dramatically reduces the effort required to build a tutor, but still leaves the most difficult task; domain authoring. We introduced WETAS-Ontology, a tool that enables ITS authors to model the domain graphically using ontology. A pilot study at an e-learning summer school showed whilst this approach did not suit everyone, some students were able to develop domain models extremely quickly using this system; one group of students developed a fully working tutoring system in around eight hours. This represents a significant leap in authoring efficiency compared to more traditional methods of tutor development.

In both the practical exercise and the project students developed simple tutors in a very short space of time. The reasons for this are threefold. First, WETAS removes all of the domain-independent authoring tasks. Second, the ontology helps the students visualize the model as they build it. Finally, the constraint generator removes the need to encode the constraints, which requires programming knowledge and can cause the author to waste time debugging errors. However, the generator has another important benefit; it removes the need for the student to test what aspects of each concept need to be tested. This has the effect of reducing the task by a factor of five, this being the average number of constraints generated per node. Unfortunately this advantage comes at a cost: the semantics that can be represented in the ontology are somewhat limited. In particular, there is no easy way to indicate that two or more solution constructs are *equivalent*. The constraint generator produces the constraints by applying simple templates to each node in the hierarchy, which test for the existence of the concept (or sub-parts of it) in both the system's ideal solution and the student's attempt. The constraints can therefore only superficially check semantics by looking for constructs in the student solution that are *identical* to those in the ideal solution. The only exception to this is that "reversible" concepts can be reversed in order, so "fish and chips" is equivalent to "chips and fish". However, in many domains there are multiple ways to solve a problem, both at the macro level (different problem-solving strategies) and in differences in the ways a student can validly articulate the same answer. The constraints need to cater for this.

This problem might be overcome in several ways. First, the ontology could be extended to allow a third type of parent-child relationship: *equivalent-instance-of*. All children with this relationship would then be considered equivalent. For example, in the search engine domain De Morgan's law holds that NOT("fish" AND "chips") is equivalent to NOT("fish") OR NOT("chips"). A "De Morgan's" concept could be modeled (using equivalent-instance-of relationships) that is used by the constraint generator to create constraints that take this equivalence into account. Second, for some domains the ontology concepts could be annotated with a truth table that indicates how sub-concepts are combined semantically. Finally, equivalence

could be inferred from examples in an additional step (e.g. [20]). We are currently investigating these possibilities. For example, the ASPIRE authoring system infers equivalence from the differences between alternative ideal solutions, with the author aiding this process where necessary by highlighting the equivalent constructs [21].

Intelligent tutoring systems are a promising tool for delivering education remotely. To date a key problem has been the effort required to build such systems. Authoring tools like WETAS and WETAS-Ontology have the potential to make widespread deployment of ITS feasible in the near future.

References

1. Koedinger, K.R., Anderson, J.R., Hadley, W.H., and Mark, M.A., *Intelligent Tutoring Goes To School in the Big City*. International Journal of Artificial Intelligence in Education, 1997. 8: p. 30-43.
2. Mitrovic, A. *Large-Scale Deployment of three intelligent web-based database tutors*. in *Proceedings of ITI*. 2006. Cavtat, Croatia. p. 135-140.
3. Koedinger, K.R., Aleven, V., Heffernan, N., McLaren, B., and Hockenberry, M. *Opening the Door to Non-programmers: Authoring Intelligent Tutor Behavior by Demonstration*. in *7th Int. Conf. Intelligent Tutoring Systems*. 2004. Maceio, Brazil: Springer-Verlag. p. 162-174.
4. Ainsworth, S.E. and Grimshaw, S., *Evaluating the REDEEM Authoring Tool: Can Teachers Create Effective Learning Environments?* International Journal of Artificial Intelligence in Education, 2004. 14(3): p. 279-312.
5. Ohlsson, S., *Constraint-Based Student Modeling*, in *Student Modeling: The Key to Individualized Knowledge-Based Instruction*, J. Greer and G. McCalla, Editors. 1994, Springer-Verlag: New York. p. 167-189.
6. Mitrovic, A. and Ohlsson, S., *Evaluation of a Constraint-Based Tutor for a Database Language.* International Journal of Artificial Intelligence in Education, 1999. 10: p. 238-256.
7. Mitrovic, A., Koedinger, K.R., and Martin, B. *A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling*. in *Ninth International Conference on User Modeling UM 2003*. 2003: Springer-Verlag. p. 313-322.
8. Martin, B. and Mitrovic, A. *WETAS: A Web-Based Authoring System for Constraint-Based ITS*. in *Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. 2002. Malaga: Springer. p. 543-546.
9. Martin, B. and Mitrovic, A. *Authoring web-based tutoring systems with WETAS*. in *International conference on computers in education*. 2002. Auckland. p. 183-187.
10. Murray, T., *Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems.* International Journal of Artificial Intelligence in Education, 1997. 8: p. 222-232.
11. Ohlsson, S., *Learning from Performance Errors.* Psychological Review, 1996. 3(2): p. 241-262.
12. Martin, B. and Mitrovic, A. *Automatic Problem Generation in Constraint-Based Tutors*. in *Sixth International Conference on Intelligent Tutoring Systems*. 2002. Biarritz: Springer. p. 388-398.
13. Martin, B. and Mitrovic, A. *Tailoring Feedback by Correcting Student Answers*. in *Fifth International Conference on Intelligent Tutoring Systems*. 2000. Montreal: Springer. p. 383-392.
14. Baghhaei, N. and Mitrovic, A. *A Constraint-based Collaborative Environment for Learning UML Class Diagrams*. in *ITS2006*. 2006. Taiwan: Springer. p. 176-186.
15. Martin, B. and Mitrovic, A. *ITS Domain Modelling: Art or Science?* in *Internation Conference on Artificial Intelligence in Education, AIED2003*. 2003. Sydney, Australia: IOS press. p. 183-190.
16. Mizoguchi, R. and Bourdeau, J., *Using Ontological Engineering to Overcome Common AI-ED Problems.* International Journal of Artificial Intelligence in Education, 2000. 11: p. 107-121.
17. Puerta, A.R. and Musen, M., *A multiple-method knowledge acquisition shell for the automatic generation of knowledge-acquisition tools.* Knowledge Acquisition, 1992. 4: p. 171-196.
18. Knublauch, H., Fergerson, R.W., Noy, N.F., and Musen, M.A. *The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications*. in *Third International Semantic Web Conference*. 2004. Hiroshima, Japan.
19. Anderson, J.R., Corbett, A.T., Koedinger, K.R., and Pelletier, R., *Cognitive Tutors: Lessons Learned.* Journal of the Learning Sciences, 1995. 4(2): p. 167-207.
20. Blessing, *A Programming by Demonstration Authoring Tool for Model-Tracing Tutors.* International Journal of Artificial Intelligence in Education, 1997. 8: p. 233-261.
21. Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., and Holland, J. *Authoring constraint-based tutors in ASPIRE*. in *ITS 2006*. 2006. Taiwan. p. 41-50.