The role of domain ontology in knowledge acquisition for ITSs

Pramuditha Suraweera, Antonija Mitrovic and Brent Martin

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

Abstract. There have been several attempts to automate knowledge acquisition for ITSs that teach procedural tasks. The goal of our project is to automate the acquisition of domain models for constraint-based tutors for both procedural and non-procedural tasks. We propose a three-phase approach: building a domain ontology, acquiring syntactic constraints directly from the ontology, and engaging the author in a dialog, in order to induce semantic constraints using machine learning techniques. An ontology is arguably easier to create than the domain model. Our hypothesis is that the domain ontology is also useful for reflecting on the domain, so would be of great importance for building constraints manually. This paper reports on an experiment performed in order to test this hypothesis. The results show that constraints sets built using a domain ontology are superior, and the authors who developed the ontology before constraints acknowledge the usefulness of an ontology in the knowledge acquisition process.

1 Introduction

Intelligent Tutoring Systems (ITS) are educational programs that assist students in their learning by adaptively providing pedagogical support. Although highly regarded in the research community as effective teaching tools, developing an ITS is a labour intensive and time consuming process. The main cause behind the extreme time and effort requirements is the knowledge acquisition bottleneck [9].

Constraint based modelling (CBM) [10] is a student modelling approach that somewhat eases the knowledge acquisition bottleneck by using a more abstract representation of the domain compared to other common approaches [7]. However, building constraint sets still remains a major challenge. In this paper, we propose an approach to automatic acquisition of domain models for constraint-based tutors. We believe that the domain ontology can be used as a starting point for automatic acquisition of constraints. Furthermore, building an ontology is a reflective task that focuses the author on the important concepts of the domain. Therefore, our hypothesis is that ontologies are also important for developing constraints manually.

To test this hypothesis we conducted an experiment with graduate students enrolled in an ITS course. They were given the task of composing the knowledge base for an ITS for adjectives in the English language. We present an overview of our goals and the results of our evaluation in this paper.

The remainder of the paper is arranged into five sections. The next section presents related work on automatic knowledge acquisition for ITSs, while Section 3 gives an overview of the proposed project. Details of enhancing the authoring shell WETAS are given in Section 4. Section 5 presents the experiment and its results. Conclusions and future work are presented in the final section.

2 Related Work

Research attempts at automatically acquiring knowledge for ITSs have met with limited success. Several authoring systems have been developed so far, such as KnoMic (Knowledge Mimic)[15], Disciple [13, 14] and Demonstr8 [1]. These have focussed on acquiring procedural knowledge only.

KnoMic is a learning-by-observation system for acquiring procedural knowledge in a simulated environment. The system represents domain knowledge as a generic hierarchy, which can be formatted into a number of specific representations, including production rules and decision trees. KnoMic observes the domain expert carrying out tasks within the simulated environment, resulting in a set of observation traces. The expert annotates the points where he/she changed a goal because it was either achieved or abandoned. The system then uses a generalization algorithm to learn the conditions of actions, goals and operators. An evaluation conducted to test the accuracy of the procedural knowledge learnt by KnoMic in an air combat simulator revealed that out of the 140 productions that were created, 101 were fully correct and 29 of the remainder were functionally correct [15]. Although the results are encouraging, KnoMic's applicability is restricted to simulated environments.

Disciple is a shell for developing personal agents. It relies on a semantic network that describes the domain, which can be created by the author or imported from a repository. Initially the shell has to be customised by building a domain-specific interface, which gives the domain expert a natural way of solving problems. Disciple also requires a problem solver to be developed. The knowledge elicitation process is initiated by a proble-solving example provided by the expert. The agent generalises the given example with the assistance of the expert and refines it by learning from experimentation and examples. The learned rules are added to the knowledge base.

Disciple falls short of providing the ability for teachers to build ITSs. The customisation of Disciple requires multiple facets of expertise including knowledge engineering and programming that cannot be expected from a typical domain expert. Furthermore, as Disciple depends on the problem solving instances provided by the domain expert, they should be selected carefully to reflect significant problem states.

Demonstr8 is an authoring tool for building model-tracing tutors for arithmetic. It uses programming by demonstration to reduce the authoring effort. The system provides a drawing tool like interface for building the student interface of the ITS.

The system automatically defines each GUI element as a working memory element (WME), while WMEs involving more than a single GUI element must be defined manually. The system generates production rules by observing problems being solved by an expert. Demonstr8 performs an exhaustive search in order to determine the problem-solving procedure used to obtain the solution. If more than one such procedure exists, then the user would have to select the correct one. Domain experts must have significant knowledge of cognitive science and production systems in order to be able to specify higher order WMEs and validate production rules.

3 Automatic constraint acquisition

Existing approaches to knowledge acquisition for ITSs acquire procedural knowledge by recording the expert's actions and generalising recorded traces using machine learning algorithms. Even though these systems are well suited to simulated environments where goals are achieved by performing a set of steps in a specific order, they fail to acquire knowledge for non-procedural domains. Our goal is to develop an authoring system that can acquire procedural as well as declarative knowledge.

The authoring system will be an extension of WETAS [4], a web-based tutoring shell. WETAS provides all the domain-independent components for a text-based ITS, including the user interface, pedagogical module and student modeller. The pedagogical module makes decisions based on the student model regarding problem/feedback generation, whereas the student modeller evaluates student solutions by comparing them to the domain model and updates the student model. The main limitation of WETAS is its lack of support for authoring the domain model.

WETAS is based on Constraint based modelling (CBM), proposed by Ohlsson [10] which is a student modelling approach based on his theory of learning from performance errors [11]. CBM uses constraints to represent the knowledge of the tutoring system [6, 12], which are used to identify errors in the student solution. CBM focuses on correct knowledge rather than describing the student's problem solving procedure as in model tracing [7]. As the space of false knowledge is much grater than correct knowledge, in CBM knowledge is modelled by a set of constraints that identify the set of correct solutions from the set of all possible student inputs. CBM represents knowledge as a set of ordered pairs of relevance and satisfaction conditions. The relevance condition identifies the states in which the constraint is relevant, while the satisfaction condition identifies the subset of the relevant states in which the constraint is satisfied.

Manually composing a constraint set is a labour intensive and time-consuming task. For example, SQL-Tutor contains over 600 constraints, each taking over an hour to produce [5]. Therefore, the task of composing the knowledge base of SQL-Tutor would have taken over 4 months to complete. Since WETAS does not provide any assistance for developing the knowledge base, typically a knowledge base is composed using a text editor. Although the flexibility of a text editor may be powerful for knowledge engineers, novices tend to be overwhelmed by the task.

Our goal is to significantly reduce the time and effort required to generate a set of constraints. We see the process of authoring a knowledge base as consisting of three phases. In the first phase, the author composes the domain ontology. This is an interactive process where the system evaluates certain aspects of the ontology. The expert may choose to update the ontology according to the feedback given by the system. Once the ontology is complete, the system extracts certain constraints directly from it, such as cardinality restrictions for relationships or domains for attributes. The second stage involves learning from examples. The system learns constraints by generalising the examples provided by the domain expert. If the system finds an anomaly between the ontology and the examples, it alerts the user, who corrects the problem. The final stage involves validating the generated constraints. The system generates examples to be labelled as correct or incorrect by the domain expert. It may also present the constraints in a human readable form, for the domain expert to validate.

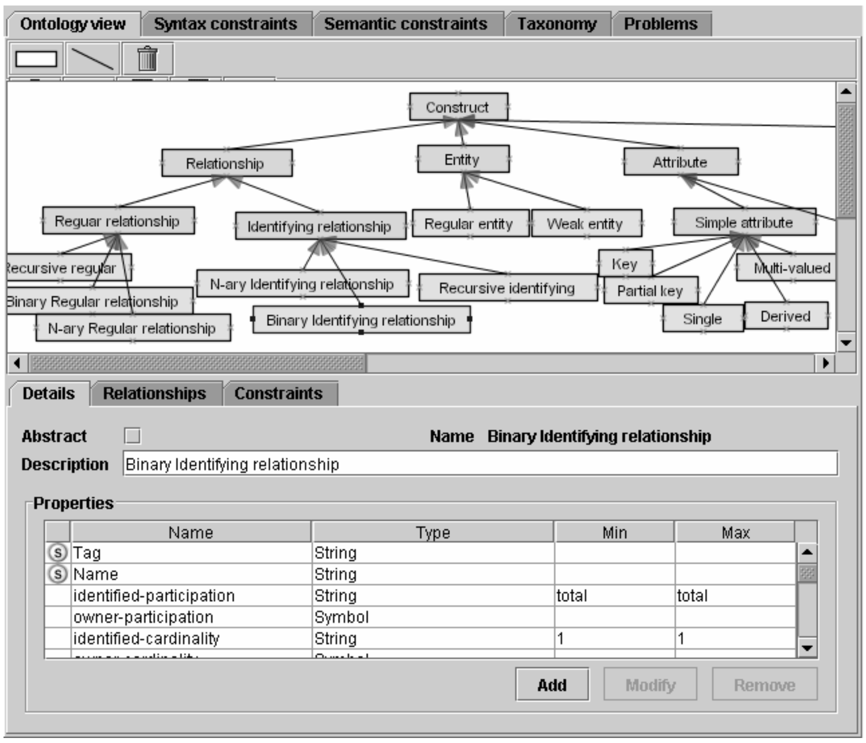
4 Enhancing WETAS: Knowledge Base Generation via Ontologies

We propose that the initial authoring step be the development of a domain ontology, which will later be used to generate constraints automatically. An ontology describes the domain, by identifying all domain concepts and relationships between them. We believe that it is highly beneficial for the author to develop a domain ontology even when the constraint sets is developed manually, because this helps the author to reflect on the domain. Such an activity would enhance the author's understanding of the domain and therefore be a helpful tool when identifying constraints. We also believe that categorising constraints according to the ontology would assist the authoring process.

To test our hypothesis, we built a tool as a front-end for WETAS. Its main purpose is to encourage the use of domain ontology as a means of visualising the domain and organising the knowledge base. The tool supports drawing the ontology, and composing constraints and problems. The ontology front end for WETAS was developed as a Java applet. The interface (**Fig. 1**a) consists of a workspace for developing a domain ontology (*ontology view*) and editors for syntax constraints, semantic constraints, macros and problems. As shown in **Fig. 1**a, concepts are represented as rectangles, and sub-concepts are related to concepts by arrows. The concept details such as attributes and relationships can be specified in the bottom section of the ontology view. The interface also allows the user to view the constraints related to a concept.

The ontology shown in **Fig. 1**a conceptualises the Entity Relationship (ER) data model. *Construct* is the most general concept, which includes *Relationship*, *Entity*, *Attribute* and *Connector* as sub-concepts. *Relationship* is specialized into *Regular* and *Identifying* ones. *Entity* is also specialized, according to its types, into *Regular* and *Weak* entities. *Attribute* is divided in to two sub-concepts of *Simple* and *Composite* attributes. The details of the *Binary Identifying relationship* concept are depicted in **Fig. 1**. It has several attributes (such as *Name* and *Identified-participation*), and three relationships (**Fig. 1**b): *Attributes* (which is inherited from *Relationship*), *Owner*, and *Identified-entity*. The interface allows the specification of restrictions of these relationships in the form of cardinalities. The relationship between *Identifying relationship* and *Regular entity* named *Owner* has a minimum cardinality of 1. The interface also allows the author to display the constraints for each concept (**Fig. 1**c). The constraints can be either directly entered in the ontology view interface or in the syntax/semantic constraints editor.

a



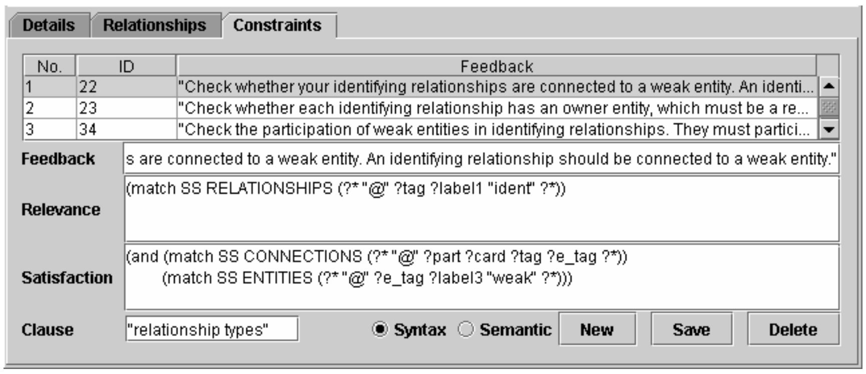
Abstract ☐ **Name** Binary Identifying relationship

Description: Binary Identifying relationship

Properties

	Name	Type	Min	Max
S	Tag	String		
S	Name	String		
	identified-participation	String	total	total
	owner-participation	Symbol		
	identified-cardinality	String	1	1

b



No.	ID	Feedback
1	22	"Check whether your identifying relationships are connected to a weak entity. An identi...
2	23	"Check whether each identifying relationship has an owner entity, which must be a re...
3	34	"Check the participation of weak entities in identifying relationships. They must partici...

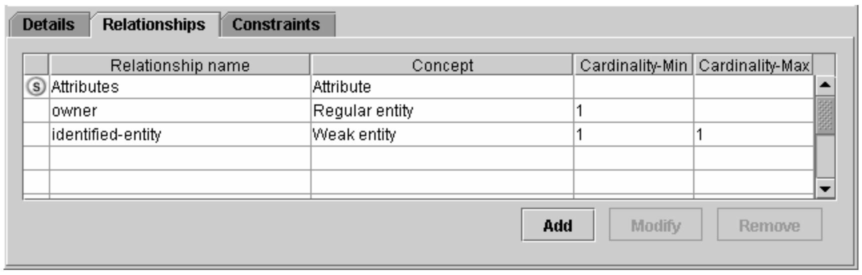
Feedback: s are connected to a weak entity. An identifying relationship should be connected to a weak entity."

Relevance:
```
(match SS RELATIONSHIPS (?* "@" ?tag ?label1 "ident" ?*))
```

Satisfaction:
```
(and (match SS CONNECTIONS (?* "@" ?part ?card ?tag ?e_tag ?*))
     (match SS ENTITIES (?* "@" ?e_tag ?label3 "weak" ?*)))
```

Clause: "relationship types" [✓ Syntax] Semantic

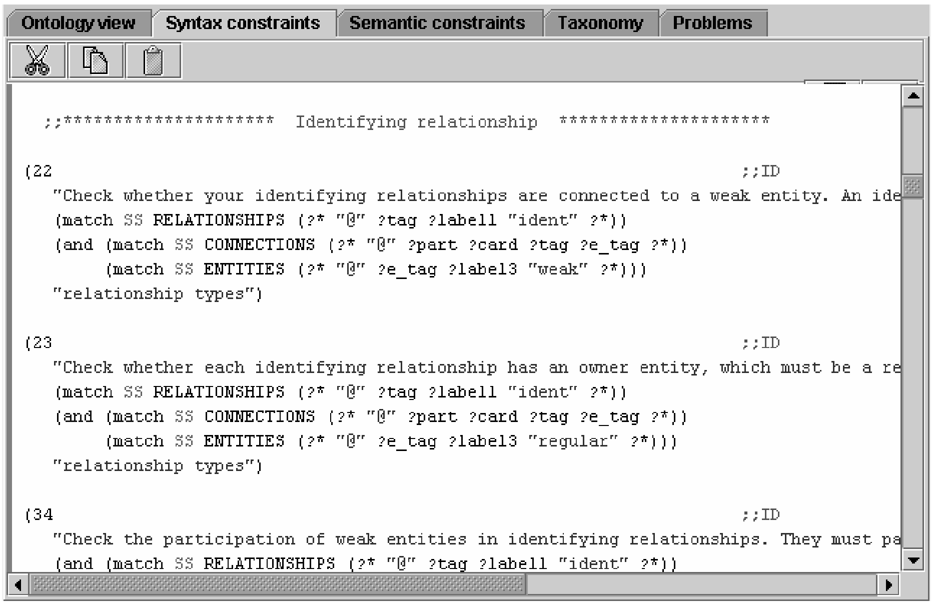
c



	Relationship name	Concept	Cardinality-Min	Cardinality-Max
S	Attributes	Attribute		
	owner	Regular entity	1	
	identified-entity	Weak entity	1	1

Fig. 1. Ontology for ER data model

The constraint editors allow authors to view and edit the entire list of constraints and problems. As shown in **Fig. 2**, the constraints are categorised according to the concepts that they are related to by the use of comments. The Ontology view extracts constraints from the constraint editors and displays them under the categorised concept. **Fig. 2** shows two constraints (Constraint 22 and 23) that belong to *Identifying relationship* concept.



```
  ;;*********************  Identifying relationship  ********************

(22                                                          ;;ID
   "Check whether your identifying relationships are connected to a weak entity. An ide
   (match SS RELATIONSHIPS (?* "@" ?tag ?label1 "ident" ?*))
   (and (match SS CONNECTIONS (?* "@" ?part ?card ?tag ?e_tag ?*))
        (match SS ENTITIES (?* "@" ?e_tag ?label3 "weak" ?*)))
   "relationship types")

(23                                                          ;;ID
   "Check whether each identifying relationship has an owner entity, which must be a re
   (match SS RELATIONSHIPS (?* "@" ?tag ?label1 "ident" ?*))
   (and (match SS CONNECTIONS (?* "@" ?part ?card ?tag ?e_tag ?*))
        (match SS ENTITIES (?* "@" ?e_tag ?label3 "regular" ?*)))
   "relationship types")

(34                                                          ;;ID
   "Check the participation of weak entities in identifying relationships. They must pa
   (and (match SS RELATIONSHIPS (?* "@" ?tag ?label1 "ident" ?*))
```

Fig. 2. Syntax constraints editor

All domain related information is saved on the server as required by WETAS. The applet monitors all significant events in the ontology view and logs them with their time stamps. The logged events include log in/out, adding/deleting concepts etc.

5 Experiment

We hypothesized that composing the ontology and organising the constraints according to its concepts would assist in the task of building a constraint set manually. To evaluate our hypothesis, we set 18 students enrolled in the 2003 graduate course on Intelligent Tutoring Systems at the University of Canterbury the task of building a tutor using WETAS for adjectives in the English language.

The students had attended 13 lectures on ITS, including five on CBM, before the experiment. They also had a 50 minute presentation on WETAS, and were given a description of the task, instructions on how to write constraints, and the section on adjectives from a text book for English vocabulary [2]. The students had three weeks to implement the tutor. A typical problem is to complete a sentence by providing the correct form of a given adjective. An example sentence the students were given was: “My sister is much ________ than me (wise).”

The students were also free to explore LBITS [3], a tutor developed in WETAS that teaches simple vocabulary skills. The students were allowed to access the “last two letters” puzzles, where the task involved determining a set of words that satisfied the clues, with the first two letters of each word being the same as the last two letters of the previous one. All domain specific components, including its ontology, the constraints and problems, were available.

Seventeen students completed the task satisfactorily. One student lost his entire work due to a system bug, and this student’s data was not included in the analysis. The same bug did not affect other students, since it was eliminated before others experienced it. Table 1 gives some statistics about the remaining students, including their interaction times, numbers of constraints and the marks for constraints and ontology.

The participants took 37 hours to complete the task, spending 12% of the time in the ontology view. The time in the ontology view varied widely, with a minimum of 1.2 and maximum of 7.2 hours. This can be attributed to different styles of developing the ontology. Some students may have developed the ontology on paper before using the system, whereas others developed the whole ontology online. Furthermore, some students also used the ontology view to add constraints. However, the logs showed that this was not a popular option, as most students composed constraints in the constraint editors. One factor that contributed to this behaviour may be the restrictiveness of the constraint interface, which displays only a single constraint at a time.

WETAS distinguishes between semantic and syntactic constraints. In the domain of adjectives, it is not clear as to which category the constraints belong. For example, in order to determine whether a solution is correct, it is necessary to check whether the correct rule has been applied (semantics) and whether the resulting word is spelt correctly (syntax). This is evident in the results for the total number of constraints for each category. The averages of both categories are similar (9 semantic constraints and 11 syntax constraints). Some participants have included most of their constraints as semantic and others vice versa. Students on average composed 20 constraints in total.

We compared the participants' solution to the "ideal" solution. The marks for these two aspects are given under *Coverage* (the last two columns in Table 1). The ideal knowledge base consists of 20 constraints. The *Constraints* column gives the number of the ideal constraints that are accounted for in the participants' constraint sets. Note that the mapping between the ideal and participants' constraints is not necessarily 1:1. Two participants accounted for all 20 constraints. On average, the participants covered 15 constraints. The quality of constraints was high generally.

The ontologies produced by the participants were given a mark out of five (the *Ontology* column in Table 1). All students scored high, as expected because the ontology was straightforward. Almost every participant specified a separate concept for each group of adjectives according to the given rules [2]. However, some students constructed a flat ontology, which contained only the six groupings corresponding to the rules (see Fig. 3a). Five students scored full marks for the ontology by including the degree (comparative or superlative) and syntax such as spelling (see Fig. 3b).

Even though the participants were only given a brief description of ontologies and the example ontology of LBITS, they created ontologies of a reasonable standard. However, we cannot make any general assumptions on the difficulty of constructing ontologies since the domain of adjectives is very simple. Furthermore, the six rules for determining the comparative and superlative degree of an adjective gave strong hints on what concepts should be modelled.

	Time (hours)		Number of constraints			Coverage	
	Total	Ontology view	Semantic	Syntax	Total	Constraints	Ontology
S1	38.16	4.57	27	3	30	20	5
S2	51.55	7.01	3	10	13	19	4
S3	10.22	1.20	14	1	15	17	4
S4	45.25	2.54	30	4	34	18	5
S5	48.96	4.91	11	5	16	20	4
S6	44.89	4.66	24	1	25	18	5
S7	18.97	2.87	1	15	16	17	4
S8	22.94	4.99	3	18	21	15	3
S9	34.29	4.30	11	4	15	18	5
S10	33.90	7.23	0	14	14	18	3
S11	55.76	3.28	16	1	17	17	5
S12	30.46	2.84	0	16	16	10	3
S13	60.94	3.47	1	15	16	13	3
S14	32.42	1.96	1	17	18	12	3
S15	33.35	4.04	1	14	15	11	3
S16	29.60	6.24	0	30	30	4	5
Mean	36.98	4.13	8.94	10.50	19.44	15.44	4.00
S.D.	13.66	1.72	10.47	8.23	6.60	4.37	0.89

Table 1. Results

Fourteen participants categorised their constraints according to the concepts of the ontology as shown in **Fig. 2**. For these participants, there was a significant correlation between the ontology score and the constraints score (0.679, p<0.01). However, there was no significant correlation between the ontology score and the constraints score when all participants were considered. This strongly suggests that the participants used the ontology to write constraints developed better constraints.

An obvious reason for this finding may be that more able students produced better ontologies and also produced a complete set of constraints. To test this hypothesis, we determined the correlation between the participant's final grade for the course (which included other assignments) and the ontology/constraint scores. There was indeed a strong correlation (0.840, p<0.01) between the grade and the constraint score. However, there was no significant correlation between the grade and the ontology score. This lack of a relationship can be due to a number of factors. Since the task of building ontologies was novel for the participants, they may have found it interesting and performed well regardless of their ability. Another factor is that the participants had more practise at writing constraints (in other assignments for the same course) than on ontologies. Finally, the simplicity of the domain could also be a contributing factor.

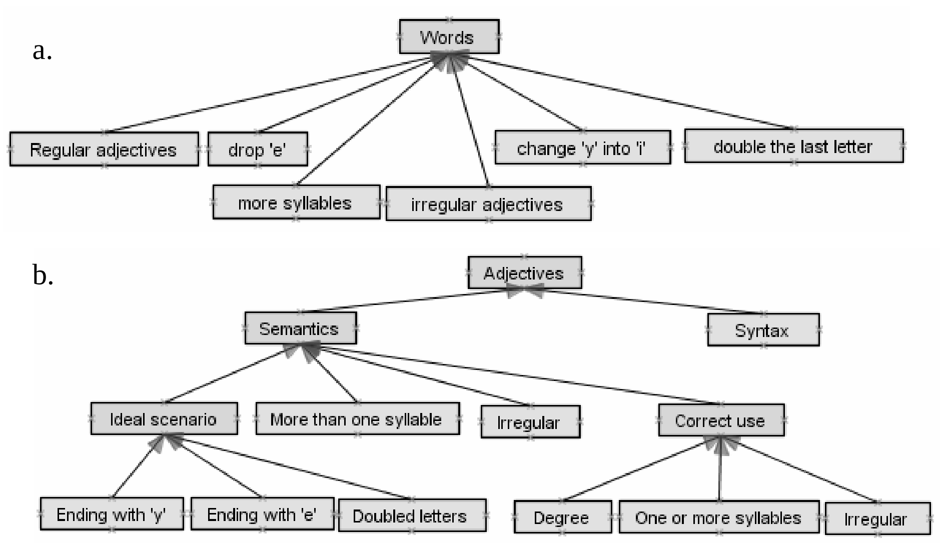


Fig. 3. Ontologies constructed by students

The participants spent 2 hours per constraint (sd=1 hour). This is twice the time reported in [8], but the participants are neither knowledge engineers nor domain experts, so the difference is understandable. The participants felt that building an ontology made constraint identification easier. The following comments were extracted from their reports: "*Ontology helped me organise my thinking;*" "*The ontology made me easily define the basic structure of this tutor;*" "*The constraints were* *constructed based on the ontology design;*" "*Ontology was designed first so that it provides a guideline for the tasks ahead.*"

The results indicate that ontologies do assist constraint acquisition: there is a strong correlation between the ontology score and the constraints score for the participants who organised the constraints according to the ontology. Subjective reports confirmed that the ontology was used as a starting point when writing constraints. As expected, more able students produced better constraints. In contrast, most participants composed good ontologies, regardless of their ability.

6 Conclusions

We performed an experiment to determine whether the use of domain ontologies would assist manual composition of constraints for constraint-based ITSs. The WETAS authoring shell was enhanced with a tool that allowed users to define a domain ontology and use it as the basis for organizing constraints. We showed that constructing a domain ontology indeed assisted the creation of constraints. Ontologies enable authors to visualise the constraint set and to reflect on the domain, assisting them to create more complete constraint bases.

We intend to enhance WETAS further by automating constraint acquisition. Preliminary results show that many constraints can be induced directly from the domain ontology. We will also be exploring ways of using machine learning algorithms to automate constraint acquisition from dialogs with domain experts.

Acknowledgements The work reported here has been supported by the University of Canterbury Grant U6532.

References

1. Blessing, S.B.: A Programming by Demonstration Authoring Tool for Model-Tracing Tutors. Artificial Intelligence in Education, 8 (1997) 233-261
2. Clutterbuck, P.M.: The art of teaching spelling: a ready reference and classroom active resource for Australian primary schools. Longman Australia Pty Ltd, Melbourne, 1990.
3. Martin, B., Mitrovic, A.: Authoring Web-Based Tutoring Systems with WETAS. In: Kinshuk, Lewis, R., Akahori, K., Kemp, R., Okamoto, T., Henderson, L. and Lee, C.-H. (eds.) Proc. ICCE 2002 (2002) 183-187
4. Martin, B., Mitrovic, A.: WETAS: a Web-Based Authoring System for Constraint-Based ITS. Proc. 2nd Int. Conf on Adaptive Hypermedia and Adaptive Web-based Systems AH 2002, Springer-Verlag, Berlin Heidelberg New York, pp. 543-546, 2002.
5. Mitrovic, A.: Experiences in Implementing Constraint-Based Modelling in SQL-Tutor. In: Goettl, B.P., Halff, H.M., Redfield, C.L. and Shute, V.J. (eds.) Proc. 4th Int. Conf. on Intelligent Tutoring Systems, San Antonio, (1998) 414-423
6. Mitrovic, A.: An intelligent SQL tutor on the Web. Artificial Intelligence in Education, 13, (2003) 171-195
7. Mitrovic, A., Koedinger, K. Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modeling. In: Brusilovsky, P., Corbett, A. and Rosis, F.d. (eds.) Proc.

UM2003, Pittsburgh, USA, Springer-Verlag, Berlin Heidelberg New York (2003) 313-322

8. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-based Tutor for a Database Language. Artificial Intelligence in Education , 10(3-4) (1999) 238-256
9. Murray, T.: Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems. Artificial Intelligence in Education, 8 (1997) 222-232
10. Ohlsson, S.: Constraint-based Student Modelling. Proc. Student Modelling: the Key to Individualized Knowledge-based Instruction, Springer-Verlag (1994) 167-189
11. Ohlsson, S.: Learning from Performance Errors. Psychological Review, 103 (1996) 241-262
12. Suraweera, P., Mitrovic, A.: KERMIT: a Constraint-based Tutor for Database Modeling. In: Cerri, S., Gouarderes, G. and Paraguacu, F. (eds.) Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002, Biarritz, France, LCNS 2363 (2002) 377-387
13. Tecuci, G.: Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies. Academic press, 1998.
14. Tecuci, G., Keeling, H.: Developing an Intelligent Educational Agent with Disciple. Artificial Intelligence in Education, 10 (1999) 221-237
15. van Lent, M., Laird, J.E.: Learning Procedural Knowledge through Observation. Proc. Int. Conf. on Knowledge Capture, (2001) 179-186