Using Theory Revision to Model Students and Acquire Stereotypical Errors *

Paul T. Baffes and Raymond J. Mooney
Department of Computer Sciences,
University of Texas at Austin,
Austin, Texas 78712
baffes@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

Student modeling has been identified as an important component to the long term development of Intelligent Computer-Aided Instruction (ICAI) systems. Two basic approaches have evolved to model student misconceptions. One uses a static, predefined library of user bugs which contains the misconceptions modeled by the system. The other uses induction to learn student misconceptions from scratch. Here, we present a third approach that uses a machine learning technique called *theory revision*. Using theory revision allows the system to automatically construct a bug library for use in modeling while retaining the flexibility to address novel errors.

1 Introduction

One of the most important components of an Intelligent Computer-Aided Instruction (ICAI) system is the student model (Wenger, 1987). Some researchers have argued (Carbonell, 1970; Laubsch, 1975) that the effectiveness of an ICAI system depends heavily upon its student modeling component. Without the flexibility to model novel student errors, ICAI systems will not progress much beyond today's electronic page turners with canned responses tuned to the average student.

Over the last two decades, several techniques for student modeling have been developed. One method, called *overlay* modeling (Carr and Goldstein, 1977), assumes a student's knowledge is always a subset of the correct domain knowledge. While simple to implement, this method is incapable of capturing misconceptions, or *bugs*, that represent faulty student knowledge.

To capture such misconceptions, other researchers (Brown and Burton, 1978; Burton, 1982; Brown and VanLehn, 1980; Sleeman and Smith, 1981) have focused on the use of bug libraries. In these approaches, models are built by matching student behavior against a catalog of bugs. Typically, such catalogs are either difficult to construct or fail to cover a wide enough range of behaviors.

A third method of student modeling attempts to model student misconceptions without overlays or a bug library (Langley et al., 1984; Ohlsson and Langley, 1985). Here, *induction* is used to construct a student model from examples of student behavior. While this provides more flexibility, in general accurate induction requires a large number of such examples. Moreover, this approach cannot take advantage of likely misconceptions which could be preprogrammed.

Here we present a new algorithm for student modeling called ASSERT (Acquiring Stereotypical Student Errors using Revision of Theories). There are two main contributions of our algorithm. First, ASSERT demonstrates a new method for constructing student models using a machine learning technique called *theory revision*. Theory revision allows ASSERT to build models more accurately with fewer examples of student behavior. Theory revision also enables ASSERT to utilize a complete or partial bug library. Second, ASSERT provides a new method for automatically constructing and extending a bug library by combining multiple student models into a *stereotypical student model*. Thus ASSERT can can create and use a bug library, while retaining the flexibility to address novel student errors.

2 Overview of Theory Revision

Theory revision algorithms modify existing rule bases to make them consistent with a given set of examples. Unlike induction algorithms which receive only examples as input, theory revision systems expect both the examples and a set of rules (theory). Typically, theory revision algorithms are used under the assumption that the rules are partially correct but not yet completely defined to cover all examples. The theory is successively refined, by specialization and generalization, until it is consistent with the examples. Most theory revision systems attempt to change the input rules as little as possible to accom-

*This research was supported by the NASA Graduate Student Researchers Program under grant number NGT-50732, the National Science Foundation under grant IRI-9102926, the NASA Ames Research Center under grant NCC 2-629, and a grant from the Texas Advanced Research Program.

modate the examples.

Unfortunately we do not have the space to describe theory revision in detail. However, note that theory revision systems have been implemented using a variety of techniques, including both logic and connectionist frameworks. Here we use the EITHER system, which revises theories expressed in an extended propositional logic. In EITHER, theories consist of rules written as Horn clauses and examples represented as vectors of observable features. EITHER uses abductive and inductive reasoning to affect six types of changes. Antecedents can be specialized, generalized, added or deleted, and rules may be added or removed from the theory. For a more detailed overview of theory revision and the EITHER algorithm, see (Ourston and Mooney, 1990).

3 The ASSERT Algorithm

3.1 Student Modeling as Theory Revision

Our description of the ASSERT algorithm begins with the observation that student modeling can be viewed as theory revision. Due to the restrictions imposed by EITHER, it is assumed that the tutoring task is a categorization problem (multiple concept lesson). While categorization problems have not been a major focus of ICAI modeling efforts, concept lessons have a well understood pedagogy (Dick and Carey, 1990) and are common CAI applications. Furthermore, as Gilmore and Self (Gilmore and Self, 1988) have pointed out, machine learning has been successfully applied in categorization domains making it natural to explore its potential in concept tutorials. Other tasks, specifically procedural ones, in general cannot be represented using propositional Horn clauses. It is important to point out, however, that the basic technique of using theory revision for student modeling is not limited to categorization domains since other theory revision algorithms may use different underlying representations.

Given this assumption, the correct knowledge for the task can be represented in a straightforward manner using Horn clause rules. Each concept is represented as the head of a Horn clause, and the components of that concept make up the predicates that form the body of the clause. Disjunctive concepts are represented using multiple clauses. Figure 1 shows part of a theory for animal classification.

The full theory classifies examples as one of twelve different animals. The rules form a hierarchy where the consequents of some rules are referenced as antecedents in others. Disjunctive concepts (*e.g.* "**mammal**") are represented by multiple rules. Examples are classified by the theory via rule chaining. For instance, the following example

```
mammal    ←  birth=live
mammal    ←  feed-young=milk
ungulate  ←  mammal & ruminate
giraffe   ←  ungulate & neck=long &
             pattern=spots
```

Figure 1: Animal classification rules.

```
(birth=live & ruminate & neck=long &
 feed-young=milk & pattern=spots)
```

is classified by the rules in Figure 1 as a giraffe. Either **birth=live** or **feed-young=milk** suffices to prove **mammal** which, when combined with **ruminate** proves **ungulate**. The rest of the facts combine for the final categorization as a giraffe.

3.2 Constructing Student Models

Modeling faulty student knowledge now becomes a matter of modifying the theory to make it consistent with responses generated by the student. In other words, the *correct* theory modeling a *perfect* student is altered using theory revision to match actual student behavior. This is contrary to the typical use of theory revision, but in principle there is no difference. We are simply reversing the notion of "goodness": instead of fixing incorrect theories, we use theory revision to *introduce* faults to model the incorrect knowledge of the student.

As an example, consider again the giraffe example above. If the feature **pattern=spots** were absent and the student still classified the example as a giraffe, there would be an inconsistency between the rules and the student's observed behavior. EITHER would modify the rules to account for the discrepancy by removing the **pattern=spots** antecedent from the giraffe rule, as long as this change remained consistent with other classifications made by the student.

ASSERT begins with a correct theory of perfect student behavior and a list of examples as categorized by a particular student. These student categorizations could be collected in any number of ways including a multiple choice test where the student classifies examples represented as lists of features (such tests are common in the instructional design of CAI systems). Here we assume that a multiple choice test can be constructed from a pool of examples and the results given to EITHER as examples of a particular student's behavior. EITHER then changes the correct input theory to match the student's erroneous classifications.

3.3 Building the Stereotypical Model

For many tutoring domains, it is possible to outline typical misconceptions that a student might exhibit. This is one of the justifications for the bug library approach described earlier. Assuming that

such errors will be common, it makes sense to collect several student models and note the commonalities that exist across students. Using these commonalities one can form a representative student model which we call a *stereotypical student model*. Construction of the stereotypical student model proceeds in four phases as follows.

Phase 1: *Collection of student models.* First, several student models must be generated from the same input theory using the process already described.

Phase 2: *Sorting of rule changes.* Next, all changes from all the student models are grouped by the rule altered and the *type* of change made. As mentioned earlier, there are six types of changes or *deviations* that EITHER can make to a rule. Each deviation may consist of multiple *component* changes to the rule. The result of this sorting is a list of proposed deviations to each rule, grouped by type. The size of the group equals the number of different student models that proposed a deviation to the given rule.

Phase 3 *Thresholding.* Each group of deviations associated with a rule is discarded if the size of that group does not exceed a desired *threshold*. This ensures that only those changes which are common to multiple students are incorporated into the stereotypical student model. The threshold can be modified to make the system more or less conservative about what deviations are considered stereotypical.

Phase 4: *Extraction of common changes.* After thresholding, all the deviations within a group represent a particular type of change to the rule. However, since these changes come from different student models, they will not necessarily be the same. To pull out only what is common among all the deviations, ASSERT uses the *common component extraction* algorithm shown in Figure 2. This algorithm measures commonality using two metrics: (1) the number of student models that contain the component and (2) the size of the component, where larger components represent more specific changes. Large frequent changes are preferred. The algorithm is iterative; as each common component is selected, that component is removed from all the deviations of the group before selecting the next component.

To illustrate the steps for constructing a stereotypical student model, refer again to Figure 1. Assume that three student models have been generated, and all have proposed changes to the last rule of the theory as shown in Figure 3. Assume further that the value of the threshold is 2. Since there are two different types of changes, two groups of deviations will be formed. The first, for adding rules, will contain all the `giraffe` rules from each student model. The second, for generalizing the mammal

1. Compare all components of all deviations.
2. For each component-component comparison, find the common subcomponent.
3. Store each subcomponent with a count of the number of *different* student models in which it was present. Call this count "N".
4. Select the "best" subcomponent based on the formula "L*N" where "L" is the length of the subcomponent. Add the subcomponent to the common deviations to be returned.
5. Remove all components from all deviations that are subsumed by the "best" subcomponent of step 4.
6. Repeat steps 1-5 until there are no common subcomponents (*i.e.*, step 2 produces the empty set). Return the subcomponents collected in step 4.

Figure 2: Common component extraction algorithm.

Student Model 1: 2 rules added
`giraffe` ← `foot-type=hoof` & `ungulate`
`giraffe` ← `color=tawny`

Student Model 2: 1 rules added, 1 rule changed
`giraffe` ← `color=tawny` & `ungulate`
`mammal` ← `birth=live or egg`

Student Model 3: 2 rules added
`giraffe` ← `foot-type=hoof` & `ungulate`
`giraffe` ← `color=tawny` & `ruminate`

Figure 3: Example student models.

rule, will contain only the `mammal` rule from student model 2. This second group will be thrown out during thresholding since only one deviation is in the group and the threshold is set at 2.

This leaves the three student models proposing added rules. Table 1 shows how each of the components of these deviations is measured by the common component extraction algorithm. While `color=tawny` and `ungulate` appear the most frequently ($N = 3$), the conjunct `foot-type=hoof & ungulate` has a larger product ($L * N = 4$) and is thus selected first.

Next, all of the rules that are subsumed by `foot-type=hoof & ungulate`, are removed from the student models. This leaves the second rule from model 1, the first rule from model 2, and the second rule from model 3. The only remaining common element is `color=tawny` which is extracted as the second component of the stereotypical model. This last extraction covers the rest of the remaining

Subconjunct	L	N	$L * N$
`foot-type=hoof`	1	2	2
`ungulate`	1	3	3
`color=tawny`	1	3	3
`foot-type=hoof & ungulate`	2	2	4

Table 1: Subconjunct comparison table.

rules. The final stereotypical model is

```
giraffe ← foot-type=hoof & ungulate
giraffe ← color=tawny
```

3.4 Using the Stereotypical Model

Once the stereotypical student model has been generated, it can be used directly as a bug library. However there are two different ways of incorporating its information into the modeling process. One method would be to modify the search mechanisms employed by EITHER to prefer bugs in the stereotypical model over the normal theory revision process. This would mirror the traditional use of bug libraries; bugs would be tried singly or combined in groups to predict student behavior. If no bug combination produced an accurate model, the normal theory revision process would be invoked.

A second method for incorporating the stereotypical model relies on the fact that theory revision is input/output compatible. Specifically, the input to theory revision (a theory) is identical in form to the output (a revised theory). Thus the bugs stored in the stereotypical student model can be used by simply incorporating them into the theory used to model subsequent students. Due to its simplicity, this was the approach taken here for our initial test of ASSERT.

Of course, it is unlikely that any one student will exhibit exactly the bugs of the stereotypical student model. The result is that the theory revision algorithm may be forced to repair bugs just introduced. On the other hand, it is rare to find a student who has no misconceptions in common with the average bugs. On the average, it was hoped that revising a stereotypical set of rules would be superior to revising a correct theory.

4 Empirical Results

4.1 Experimental Design

Two hypotheses formed the basis of our testing methodology. First, we expected theory revision to be more accurate at student modeling than inductive modelers due to the extra information available in the input rules. Second, we expected revising stereotypical theories to be more effective than revising correct theories since common student errors are part of the stereotypical model.

For the preliminary experiments presented here, we chose to work with artificial data for the animal classification domain referenced earlier (see Figure 1). We are currently planning experiments using actual student data collected with a CAI system for more realistic testing. As an initial domain, the animal classification rules represent a rich enough task to test our hypotheses on a variety of potential student misconceptions.

Our tests were run from a pool of 180 examples randomly generated using the correct animal classification rules (15 examples for each of the 12 categories). Artificial students were generated by making modifications to the correct theory. As each student theory was formed, it was used to relabel the 180 examples to simulate the behavior of that student. These relabeled examples act as "answers" the student would generate to the 180 "multiple choice questions."

Modifications made to the correct theory to create students were of two types. One set of modifications was predefined, with a given probability of occurrence. These simulated common errors that occurred in the student population. We used four common deviations, each with a 0.75 probability of occurrence. Two of these deleted antecedents from rules, one added an antecedent, and one changed an antecedent. To simulate individual student differences, each student theory was further subjected to random antecedent modifications with a probability of 0.10.

ASSERT was tested against both normal theory revision and induction using a two-phased approach. The first phase was used to build a stereotypical model for the second phase as follows:

1. First, 20 artificial students were created using the methods described above.
2. For each student, all 180 examples were relabeled using the student's buggy theory.
3. From these 20 students, 20 student models were generated using EITHER on all 180 relabeled examples.
4. A stereotypical student model was then built from the 20 student models using the algorithm from section 3.3 with a threshold of 10 (*i.e.*, half the students had to exhibit a bug for it to be considered "common").

Three of the four common predefined bugs ended up in the stereotypical student model. The fourth was more difficult for EITHER to generate, since it required a deletion of an antecedent followed by an addition of a different antecedent. This fourth bug ended up as two different rules in the stereotypical student model. All four resulting deviations were

Initial Rules	*modeling time (sec.)*	*test set accuracy*
stereotypical	72	96%
correct	124	84%
stereotypical, no revision	n/a	67%
correct, no revision	n/a	61%
none (induction)	12	52%

Table 2: Effect of initial rules on modeling.

applied to the correct rules to form a stereotypical student theory.

For the second phase, additional artificial students were generated to test EITHER using various initial theories. A series of experiments were run starting EITHER with (1) the correct animal rules, (2) the stereotypical student rules from phase 1 above, or (3) no initial theory. With no initial theory, EITHER defaults to an inductive learning process which uses the ID3 (Quinlan, 1986) algorithm. This phase ran as follows:

1. First, 10 new artificial students were generated using the same techniques used in phase 1. For each, the 180 examples were relabeled using the student's buggy theory.
2. 50 examples were randomly chosen from the 180 relabeled by the student as training examples. Each new student was modeled using EITHER with the same 50 examples and one of the three initial theories described above.
3. The other 130 examples were reserved for testing the accuracy of each student model as follows. Recall that the output of EITHER is a revised theory representing the student model. This theory was used to label each of the 130 test examples. These labels were compared to those generated using the student's buggy theory from step 1 to compute a percentage accuracy.

Table 2 compares the average accuracy and modeling times of EITHER started with each of the three different initial theories. For comparison purposes, we also measured the accuracy of both the correct and stereotypical theories. Statistical significance was measured using a Student t-test for paired difference of means at the 0.05 level of confidence (*i.e.*, 95% certainty that the differences were not due to random chance). All the differences shown in table 2 are statistically significant.

4.2 Discussion of Results

Both of our hypotheses were borne out by the results presented in table 2. First, it is apparent that theory revision is superior to induction in terms of accuracy. This is not surprising since induction must model *correct* as well as buggy student behavior, whereas theory revision need only alter correct rules to capture the misconceptions. The difference is even more pronounced when theory revision proceeds from the stereotypical model. Induction simply has more work to do.

Second, our results show that theory revision models students faster and more accurately when given an initial rule base that approximates typical student errors. Since all the students were generated using the same criteria, providing EITHER with the stereotypical rules effectively gives it a head start over the correct theory.

Running the correct and stereotypical theories without revision also produced interesting results. Both outperformed induction, further illustrating the disadvantage of trying to model students from scratch using a small number of examples. It is also apparent that revision is essential to effective modeling, even if the initial rules model buggy student behavior. Without revision, modeling novel student errors is simply not possible since a static library of bugs will not contain the needed information.

5 Related Work

There are two systems directly related to the work described here. Both make use of machine learning techniques to dynamically model student behavior.

Sleeman describes an extension to his PIXIE system, which models arithmetic errors, called INFER* (Sleeman et al., 1990) INFER* starts with a library of known bugs and induces rules to fill gaps between *one* student's solution and the correct rules known to the system. INFER* also relies heavily upon domain-dependent heuristics for controlling its search, and it is not clear that these techniques can be used for domains other than arithmetic. Furthermore, INFER* does not make any attempt to generalize across students in an effort to extend its library of bugs.

Like INFER*, the theory revision techniques used by ASSERT can be biased with specific heuristics and known bugs, but will operate effectively without them. Furthermore, ASSERT contains an algorithm for extracting common elements from multiple student models and thus can automatically extend its library of buggy rules.

Langley et. al. (Langley et al., 1984; Ohlsson and Langley, 1985) describe the ACM system which uses induction to generate a production system model of an individual student from a problem space of operators describing the domain. While ACM is a domain independent algorithm, Langley et. al. make the assumption that student errors are only the result of *correct* actions taken in an

incorrect *context*. This prohibits ACM from modeling illegal actions. Also, each run of ACM starts with no knowledge of when operators should be applied, forcing it to spend time modeling both correct and buggy student control knowledge that could be preprogrammed. Finally, there is no facility within ACM for building in typical student bugs nor for using the output of one run to aid subsequent modeling efforts.

6 Future Work

There are two chief disadvantages to the current ASSERT system. First, we have not tested ASSERT on real student data. Our current efforts are focused on obtaining data to run such tests. Second, as discussed above (section 3.4), our simple method of incorporating all bugs from the stereotypical model should be replaced with a revised theory revision algorithm that is biased towards preferring the bugs. Common misconceptions would be tried before more general purpose revisions so that bugs would only be considered for students who actually exhibit problems. Finally, ASSERT could also be extended to use a first-order theory revision algorithm (Richards and Mooney, 1991). This might enable ASSERT to model relational and procedural problem domains.

7 Conclusions

This paper has described a new algorithm for student modeling called ASSERT. ASSERT uses theory revision to dynamically construct student models. Multiple student models are combined to automatically construct a bug library of stereotypical student errors. Theory revision has been shown to be more effective than static bug library approaches as well as inductive modeling techniques. Revising a rule base of stereotypical student errors allows ASSERT to build and refine a bug library while retaining the flexibility to address novel misconceptions.

8 Acknowledgments

The authors would like to thank Stellan Ohlsson for his detailed insights and comments which have helped clarify this presentation.

References

Brown, J. S. and Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–192.

Brown, J. S. and VanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills. *Cognitive Science*, 4:379–426.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In Sleeman, D. H. and Brown, J. S., editors, *Intelligent Tutoring Systems*. London: Academic Press.

Carbonell, J. R. (1970). Mixed-initiative man-computer instructional dialogues. Technical Report BBN Report No. 1971, Cambridge, MA: Bolt Beranek and Newman, Inc.

Carr, B. and Goldstein, I. (1977). Overlays: a theory of modeling for computer-aided instruction. Technical Report A. I. Memo 406, Cambridge, MA: MIT.

Dick, W. and Carey, L. (1990). *The systematic design of instruction*. Glenview, IL: Scott, Foresman/Little, Brown Higher Education. Third edition.

Gilmore, D. and Self, J. (1988). The application of machine learning to intelligent tutoring systems. In Self, J., editor, *Artificial Intelligence and Human Learning*, chapter 11. New York, NY: Chapman and Hall.

Langley, P., Ohlsson, S., and Sage, S. (1984). A machine learning approach to student modeling. Technical Report CMU-RI-TR-84-7, Pittsburgh, PA.: Carnegie-Mellon University.

Laubsch, J. H. (1975). Some thoughts about representing knowledge in instructional systems. In *Proceedings of the Fourth International Joint conference on Artificial intelligence*, pages 122–125.

Ohlsson, S. and Langley, P. (1985). Identifying solution paths in cognitive diagnosis. Technical Report CMU-RI-TR-85-2, Pittsburgh, PA.: Carnegie-Mellon University.

Ourston, D. and Mooney, R. (1990). Changing the rules: a comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 815–820. Detroit, MI.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Richards, B. and Mooney, R. (1991). First-order theory revision. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 447–451. Evanston, IL.

Sleeman, D., Hirsh, H., Ellery, I., and Kim, I. (1990). Extending domain theories: two case studies in student modeling. *Machine Learning*, 5:11–37.

Sleeman, D. H. and Smith, M. J. (1981). Modelling students' problem solving. *Artificial Intelligence*, 16:171–187.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.