Revising Deductive Knowledge and Stereotypical Knowledge in a Student Model

XUEMING HUANG, GORDON I. MCCALLA, JIM E. GREER, AND ERIC NEUFELD
Department of Computational Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada S7N 0W0,
huang@skorpio.usask.ca

(14 August 1990; in final form 29 October 1990)

Abstract. A user/student model must be revised when new information about the user/student is obtained. But a sophisticated user/student model is a complex structure that contains different types of knowledge. Different techniques may be needed for revising different types of knowledge. This paper presents a student model maintenance system (SMMS) which deals with revision of two important types of knowledge in student models: deductive knowledge and stereotypical knowledge. In the SMMS, deductive knowledge is represented by justified beliefs. Its revision is accomplished by a combination of techniques involving reason maintenance and formal diagnosis. Stereotypical knowledge is represented in the Default Package Network (DPN). The DPN is a knowledge partitioning hierarchy in which each node contains concepts in a sub-domain. Revision of stereotypical knowledge is realized by propagating new information through the DPN to change default packages (stereotypes) of the nodes in the DPN. A revision of deductive knowledge may trigger a revision of stereotypical knowledge, which results in a desirable student model in which the two types of knowledge exist harmoniously.

Key words: user/student model revision, deductive knowledge, stereotypical knowledge, reason maintenance, diagnosis, default package network

1. Introduction

An intelligent tutoring system (ITS) usually has a knowledge base containing its knowledge (or beliefs) about the student called the *student model* (Sleeman and Brown, 1982; Wenger, 1987). The tutoring system can obtain this knowledge by analysing the student's responses to the system. This is called *student knowledge analysis.*[1] In fact, most research in student modeling focuses on the task of student knowledge analysis (Clancey, 1986; Wenger, 1987). However, a student model would be more useful if the information extracted during analysis could be recorded and re-used in succeeding interactions. This paper addresses the task of representing knowledge about the student and maintaining the student model, or the task of *student model*

[1] Student knowledge analysis is normally referred to as diagnosing in the ITS literature. We rename it to avoid confusion since a component of SMMS is also called the diagnostic system.

User Modeling and User-Adapted Interaction 1: 87–115, 1991.

management (McCalla et al., 1988). In particular, we focus on the issue of revising the student model when the tutoring system obtains new beliefs about the student.

The task of revising a student model would be simple if all beliefs in the student model were obtained by analysing the student's behavior (we call these analyses *observations*), since then there would be no data dependency among the beliefs, or at least we could assume that there is no such data dependency. Revision can be done by trivially adding or deleting corresponding beliefs in the student model, or by increasing or decreasing credibility of the corresponding beliefs (Burton and Brown, 1982; Kimball, 1982; Clancey, 1987). However, usually a tutor's knowledge about a student obtained from observations (by the student knowledge analysing system) is very limited. This limitation is magnified in an intelligent tutoring system because of the narrow input channel of the computer. One way to augment the knowledge is to install some deductive inference rules in the system. By applying these rules to the existing student model the system can (internally) generate new knowledge about the student. For example, assume that a mathematics tutoring system has an inference rule stating that a student who knows subtraction must also know addition (represented by a logic implication rule *knows(sub)* $\supset$ *knows(add)*). Now if it believes that the student knows subtraction (i.e., *knows(sub)* is true), then it could infer that the student also knows addition (i.e., *knows(add)* is also true). This approach has been used in some student modeling systems and user modeling systems (Sleeman, 1985; Kass and Finin, 1987; Kobsa, 1990). Generally, we refer to knowledge obtained from observations and its augmentation by deductive inferences, as *deductive knowledge.*

Although deductive knowledge is an augmentation of knowledge from direct observation, usually it is still insufficient. People make many default assumptions about others' beliefs during a dialogue. In particular, during a tutoring interaction, a tutor must make many assumptions about the student's knowledge to design advice to the student. There may be many different types of assumptions in user/student modeling (Wahlster and Kobsa, 1989). Stereotypical assumptions are one of the most important types (Rich, 1979). A *stereotype* is a package of defaults about a certain group of users. Default assumptions about a user are stored in a stereotype that models the group to which that user belongs. *Stereotypical knowledge* is important for user/student modeling because it provides a vast amount of knowledge about the user/student based merely on evidence for membership in a certain group(s).

Unlike revision of independent beliefs, revision of deductive knowledge and revision of stereotypical knowledge are difficult. These difficulties and related problems have been extensively studied by research in a theoretical AI area known as *belief revision* (Doyle, 1979; de Kleer, 1986; Martins and

Shapiro, 1988; Makinson, 1985). However, the issue of how to apply techniques of belief revision to user/student model revision has not been studied. On the other hand, although stereotypes are widely used in user modeling systems (Rich, 1979; Finin, 1989; Chin, 1989), *revision* of stereotypical knowledge remains difficult.[2] Our present goal is to investigate revision of these two types of knowledge in a student model, as well as the relationship between the two revision processes.

The paper presents the *student model maintenance system (SMMS)* shown in Figure 1. During interaction with the student, the *student knowledge analysing system (SKAS)* analyses the student's responses to generate the tutoring system's new beliefs about the student. New beliefs generated by such "observations" are sent to the SMMS which then revises the student model to accommodate the new beliefs. The updated information about the student in the student model, such as whether the student knows a particular concept or believes a particular misconception, is provided for other components of the tutoring system, including the SKAS, whenever they query the SMMS.

The student model consists of two knowledge bases. The deductive knowledge base contains knowledge generated from "observations" and knowledge generated by applying the inference rules. The stereotypical knowledge base contains knowledge in the active stereotypes. (The stereotype hierarchy of SMMS is discussed in Section 3.) Since deductive knowledge comes from a more concrete information source than stereotypical knowledge does, the former may override the latter when they are in conflict. Using the terminology of default reasoning, deductive knowledge contains "facts", while stereotypical knowledge contains "defaults" (Reiter, 1980; Finin, 1989).

A revision occurs when a set of new beliefs is generated by the SKAS and sent to the SMMS. The SMMS enters these new beliefs into its deductive knowledge base and makes the necessary revisions to maintain consistency of the deductive knowledge base. It then checks its stereotype hierarchy. If the activating conditions of some deactivated stereotypes or the retraction conditions of some activated stereotypes are satisfied due to revision of deductive knowledge, then corresponding stereotypes would be activated or retracted, thus revising the stereotypical knowledge base. This results in a consistent student model.

The rest of the paper is organized as follows: Section 2 discusses revision of deductive knowledge. Section 3 discusses revision of stereotypical knowledge. A comparison of the SMMS with related work is given in Section 4. Section 5 concludes the paper. Note that although the research is reported in the context of student modeling, the issues studied pertain to general user modeling to a greater or lesser degree.

[2] A recent paper (Kobsa, 1990) desribes a user modeling shell called BGP-MS that tackles this difficult problem.

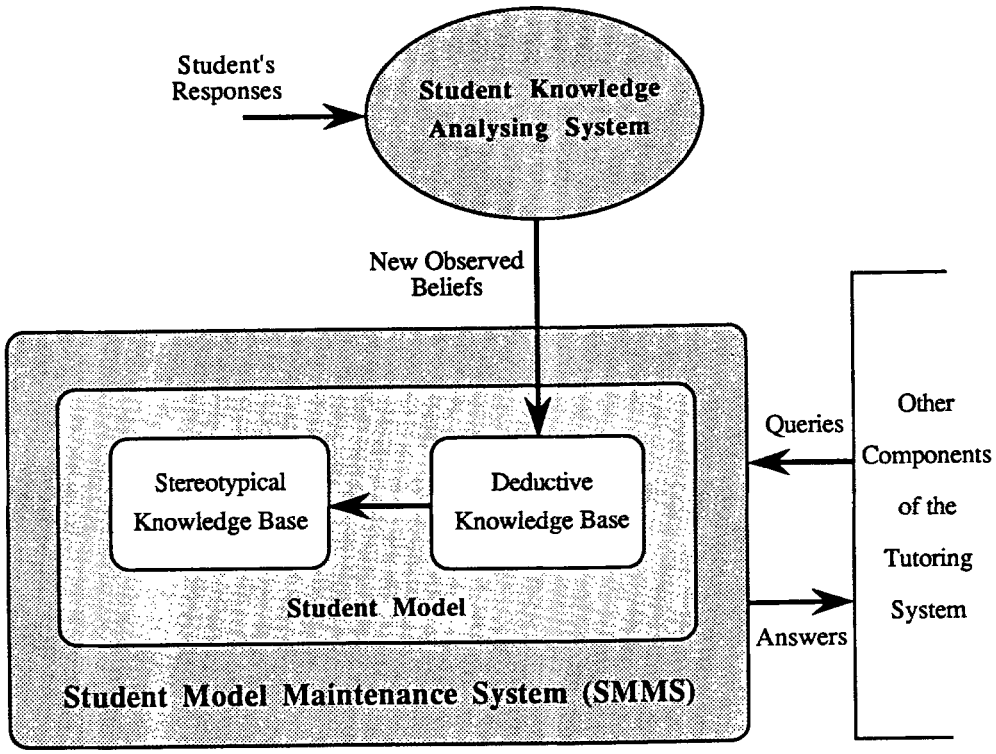

Fig. 1. SMMS in an Intelligent Tutoring System

2. Revision of Deductive Knowledge: An Evolutionary Process

In this section we discuss how techniques developed in the area of belief revision can be used for revising deductive knowledge in a student model. As a result, a system called the *evolutionary belief revision system (EBRS)* is developed to accomplish the desired revision.

2.1. Coherence Belief Revision vs. Foundations Belief Revision

There are two basic approaches to revision of deductive knowledge. One is called *coherence belief revision*, and the other called *foundations belief revision*. A common goal of both approaches is maintaining consistency of the knowledge base in response to new information. A fundamental difference between the two approaches is on the issue of whether justifications of beliefs should be taken into account during belief revision. Coherencists focus on minimal change to maintain logical consistency of the belief base, regardless of the justifications (Alchourron et al., 1985; Dalal, 1988; Gardenfors, 1990). Foundationalists insist that all beliefs must be well justified, namely,

each belief must be either directly assumed by the system (beliefs obtained from observations belong to this group) or supported by other justified beliefs (Doyle, 1979; Martins and Shapiro, 1988). Thus, a belief that has no valid justification must be removed, no matter whether or not it logically contradicts any other beliefs in the belief base.

In our early example of the inference of a mathematics tutor, the tutor derives the belief *knows(add)* from its beliefs *knows(sub)* and *knows(sub)* $\supset$ *knows(add)*. Thus, the latter two beliefs justify the former one. If later the tutor observes that the student does not know subtraction, then a new belief $\neg$ *knows(sub)* is added to the belief base, which forces removal of an old belief *knows(sub)*. Both coherencists and foundationalists agree on this point. The controversial problem is, however, whether another belief, *knows(add)*, should be removed as well. The coherence theory argues that it should stay since the knowledge base after removal of *knows(sub)*, namely {*knows(sub)* $\supset$ *knows(add)*, *knows(add)*, $\neg$ *knows(sub)*}, is already consistent. Removing *knows(add)* violates the principle of minimal change. But the foundations theory says that *knows(add)* should be removed, since it no longer has any valid justification.

Which approach is more appropriate? Although the argument into this problem remains inconclusive, many philosophers and AI researchers tend to believe that the foundations approach models how people ought to revise their beliefs, while the coherence approach models what people usually do in such situations (Harman, 1986; Ross and Anderson, 1982; Gardenfors, 1990). This implies that the approach chosen depends on the kinds of beliefs we are trying to model. The SMMS described here uses a foundations belief revision system.

2.2. ATMS AND DIAGNOSIS

Reason maintenance systems (RMS's) (Doyle, 1979; de Kleer, 1986; Martins and Shapiro, 1988) are usually considered implementations of foundations belief revision. An RMS is usually used to assist a problem solver. By recording data dependencies (e.g., justifications) and inconsistent belief spaces, the RMS guides the problem solver to work in consistent and well justified belief spaces. However, in revising a belief base such as a student model, solely ensuring consistency is insufficient: the revision should not radically change the belief base. Most existing RMS's provide little information about how to minimally modify an old belief base to accommodate new beliefs. They do not deal with the *culprit selection problem*, the problem of how to select a subset of beliefs, among many possible subsets, to remove to maintain consistency of the belief base (Martins and Shapiro, 1988).

A sub-system of SMMS, called the *Evolutionary Belief Revision System (EBRS)*, accomplishes foundations revision of deductive knowledge by com-

bining a diagnostic system (de Kleer and Williams, 1987; Reiter, 1987) with an RMS, namely a modified ATMS (de Kleer, 1986), to achieve the minimal change property. The ATMS records data dependencies and contradictions, providing the collection of inconsistent belief spaces for the diagnostic system. The diagnostic system then uses this information to select a minimal subset of beliefs and to remove the subset from the belief base. Below we give a brief sketch of the ATMS and some concepts used in diagnostic systems, primarily to establish terminology. Then we discuss the EBRS.

2.2.1. ATMS

An ATMS-based problem solver usually consists of two components: a problem solver and an ATMS. The ATMS serves as an intelligent cache for the problem solver. First, the problem solver designates a set of ATMS *nodes* (data structures of the ATMS) to be assumptions (in what follows we will call both a node and the proposition stored in the node an assumption when no confusion can be caused). An assumption is presumed to be true over the period of solving the problem, unless there is evidence to the contrary. Then the problem solver derives new beliefs from old ones, starting with the set of assumptions (which are special beliefs), and continuing until the problem solution is found. Each new belief is also assigned an ATMS node, and the set of antecedent nodes used in the derivation is recorded as a *justification* of the new belief.

An *environment* is a subset of assumptions. An ATMS node n *holds* in environment E if n can be derived from E using justifications of the nodes in the current knowledge base. An environment is *inconsistent* if a node representing a contradiction holds in it, otherwise it is *consistent.* Inconsistent environments (called *nogoods*) are recorded in a database. A consistent environment is *minimal* with respect to a node n if and only if n holds in it and in no proper subset of it. In an ATMS, in addition to the justifications, the set of minimal consistent environments, called the *label,* is also recorded in the node. If a node holds in an environment, then it also holds in all supersets of the environment. Thus, the label represents the whole environment space in which the node holds. With the label, the query as to whether a node n holds in an environment E can be quickly answered.

When the problem solver makes a new inference, it creates a new justification for the consequent node using the set of antecedent nodes of the inference, which awakes the ATMS to carry out a process of *label updating* to accommodate the new information. Label updating propagates over the knowledge base via justifications of nodes, changing the consistent environment space of each node in the knowledge base.

2.2.2. The Diagnostic Problem

Assume that one is first given a description of a system and then an observation of the system's behavior that conflicts with the expected behavior of the system. The diagnostic problem is to determine the abnormal components of the system which cause the conflicts. The concepts defined below follow de Kleer and Williams (1987).

In a diagnostic system, a *symptom* is an inconsistency detected by a higher level reasoning system. An *assumption* is a proposition that describes the normal behavior of a component of the system. A *conflict* is a set of assumptions from which a symptom can be derived. A conflict is minimal if no proper subset of it is also a conflict. A *candidate* of the diagnosis is a set of assumptions such that by removing the set, the system becomes consistent. A candidate is minimal if no proper subset of it is also a candidate. Any superset of a conflict is a conflict, and any superset of a candidate is a candidate. Therefore, the conflict space and the candidate space can be represented by the set of minimal conflicts and the set of minimal candidates, respectively. The goal of a diagnostic process is to find the set of minimal candidates. This usually requires recognition of the set of minimal conflicts first.

In a belief revision system, if we view a contradiction in the belief base as a symptom, then a set of assumptions that eventually derives a contradiction is a conflict. The problem in belief revision of finding the minimal changes of the old belief base so that the updated belief base is consistent with the new beliefs is thus reduced to the problem of finding the minimal candidates in diagnosis. This is our basic idea of using a diagnostic system in belief revision.

2.3. Basic Concepts of the EBRS

As mentioned in Section 1, deductive knowledge of the student model comes from "observations" by the SKAS and deductive inferences over the existing knowledge. Information in the deductive knowledge base is represented by propositional formulas recorded in *EBRS nodes*. In particular, contradictions discovered (denoted by $\perp$) are recorded in some distinguished EBRS nodes called *contradiction nodes*. The propositions believed by the system are called the *system's beliefs* (or *beliefs* for short). Beliefs considered true without depending on other beliefs are called *base beliefs* (akin to assumptions in the ATMS). Two kinds of beliefs are treated as base beliefs: the beliefs obtained from observations and the inference rules. The beliefs that are derived from the inferences are called *derived beliefs*.

The belief revision procedure is invoked after each observation or inference so that the new information is merged into the knowledge base which is then adjusted to accommodate the new information. This is called a *revision*

session. The *belief set* (the set of the system's beliefs) is updated in each revision session. The EBRS uses a modified ATMS to maintain consistency of its knowledge base. An ATMS is useful since it records data dependencies among beliefs and filters out inconsistent environments. However, the original ATMS is oriented towards finding all solutions in a problem solving process. It simultaneously works on all self-consistent environments (but the union of them may not be consistent). Thus, the ATMS has no concept of the system's beliefs. In the EBRS, we use the set of base beliefs from which all beliefs are derived to represent the system's beliefs. This set of base beliefs is called the *system's environment.* A subset of the system's environment is called an *active environment.* Thus, there are three kinds of environments in the EBRS (in contrast to two kinds in the ATMS): inconsistent environments, consistent environments, and active environments (which are also consistent). A proposition is currently believed if and only if the label of its EBRS node contains an active environment.

2.4. Accommodating New Information

Belief revision occurs when a set of beliefs is generated by an observation of the SKAS or by a deductive inference made by the EBRS. The EBRS puts these *just-generated* beliefs into its knowledge base. Note that a just-generated belief may not be a new belief. It may have been generated in some previous observations/inferences. The EBRS creates a new node for a just-generated belief only if the belief is new, but it usually adds a new justification to the node of the just-generated belief to record the new data dependency. If the just-generated belief is a base belief, then the justification contains only the belief itself. If it is a derived belief, then the justification contains all the beliefs and the rules used in the derivation. After a justification is added, the label updating procedure described in (de Kleer, 1986) is invoked to propagate the effects of the new information, generating an updated label for each EBRS node in the knowledge base.

The just-generated beliefs may conflict with old beliefs in the belief set, bringing contradictions into the knowledge base. Some contradictions were discovered and removed from the belief set in some previous revision session. They arise again since they get new supports after label updating. This kind of contradiction can be discovered by checking whether there is an active environment in the label of the contradiction nodes in the knowledge base. The other contradictions are new and discovered when a just-generated belief p is the negation of an old belief $\neg p$. The EBRS creates a new node to record each newly discovered contradiction. (Note that the EBRS may not discover all contradictions in the beliefs. The EBRS used here discovers a contradiction when a proposition and its negation both are believed, but discovered beliefs may be re-defined in other applications.)

Once contradictions are detected and recorded, the EBRS updates the system's *belief set* in two steps. First, all new beliefs are added into the belief set. This is done by simply putting all new base beliefs into the system's environment. The second step, removing all contradictions discovered from the system's belief set, is more difficult. This requires selecting a subset of base beliefs (called the *obsolete base belief set*) such that once the subset is removed from the system's environment, all contradictions in the belief set would be removed. There are, however, usually many such subsets (called *candidates*) in the system's environment. One way to filter out many unlikely candidates is to identify *minimal candidates*. If by removing a subset S we can remove all contradictions, then there is little reason to remove a proper superset of S instead (Gardenfors, 1984; Harman, 1986). Thus, the obsolete base belief set is chosen from the minimal candidates. For this reason, we call the revision accomplished by EBRS the *evolutionary belief revision*, in contrast to the *revolutionary belief revision* discussed in Section 3. The next section discusses a procedure in the EBRS that removes the obsolete base belief set from the system's environment. We now present an example, using beliefs about a student's knowledge of Lisp programming, to clarify what we have done so far.

In the example, we use propositional letters (e.g., A, B, ...) to represent a belief of the tutor about the student such as "The student believes that the Lisp function *car* returns a list containing the first element of the given list" (a misconception) or "The student knows the concept of recursion". Negation of a proposition (e.g., ¬A, ...) represents a negative belief such as "The student does not know the concept of recursion", rather than a disbelief such as "It is not the case that the student knows the concept of recursion." The latter is handled by the assumption that everything that is not derivable from the student model is not believed by the tutor. Inference rules such as "if the student knows the function *mapcar*, then she/he must also know the function *car*", "if the student believes that *append* is the same function as *list* (a misconception), then she/he must not know that *append* requires lists as its arguments" are represented by logic implication rules such as S ⊃ T and U ⊃ ¬V.

Assume that the SKAS has obtained the beliefs A and B in the first observation O_1. Then the EBRS applies the inference rule A ⊃ S to derive S, the rule B ∧ S ⊃ ¬ T to derive ¬ T, the rule S ⊃ U and the rule A ∧B ⊃ U to derive U, and the rule B ∧ U ⊃ V to derive V. At this moment (time t_1), the deductive knowledge base contains the following EBRS nodes (like an ATMS node, an EBRS node is of the form: [assertion, {label}, {justifications}]):

1. [A, {{1}}, {(1)}]
2. [B, {{2}}, {(2)}]
3. [A ⊃ S, {{3}}, {(3)}]
4. [B∧S ⊃ ¬T, {{4}}, {(4)}]
5. [S ⊃ U, {{5}}, {(5)}]
6. [A∧B ⊃ U, {{6}}, {(6)}]
7. [B∧U ⊃ V, {{7}}, {(7)}]
8. [S, {{1, 3}}, {(1, 3)}]
9. [T, {{1, 2, 3, 4}}, {(2, 4, 8)}]
10. [U, {{1, 3, 5}, {1, 2, 6}}, {(5, 8), (1, 2, 6)}]
11. [V, {{1, 2, 3, 5, 7}, {1, 2, 6}}, {(2, 7, 10)}]

Here, nodes 1–7 are base beliefs, while nodes 8–11 are derived beliefs. The system's environment is {1, 2, 3, 4, 5, 6, 7}. Note that every node in the knowledge base has an active environment (a subset of the system's environment) in its label. This reflects the fact that at t_1 all EBRS nodes in the knowledge base are believed. Users of the student model (usually other components of the tutoring system) are usually interested in the subset of beliefs about which specific concepts/misconceptions the student believes. We use *SDB* (for Specific Deductive Beliefs) to denote this subset. SDB contains factual knowledge about the student, excluding inference rules. Thus, at t_1 it contains beliefs in nodes 1, 2, 8, 9, 10 and 11:

$$\text{SDB}(t_1) = \{\text{A, B, S, } \neg \text{ T, U, V}\}.$$

Next the SKAS makes the second observation O_2 from which three new beliefs, C, T and ¬ U, are obtained. Three nodes are then created to record the new belief set:

12. [C, {{12}}, {(12)}]
13. [T, {{13}}, {(13)}]
14. [¬ U, {{14}}, {(14)}]

There is not much for the label updating procedure to do here because there are no data dependencies between the new beliefs and the old ones. (For examples of label updating see de Kleer, 1986.) Since all three new beliefs are base beliefs, they are entered into the system's environment which then becomes {1, 2, 3, 4, 5, 6, 7, 12, 13, 14}. Two contradictions among beliefs, (¬T, T) and (U, ¬U), are discovered. They are recorded in the contradiction nodes:

cont-1: [⊥, {{1, 2, 3, 4, 13}}, {(9, 13)}]
cont-2: [⊥, {{1, 3, 5, 14}, {1, 2, 6, 14}}, {(10, 14)}].

To remove these contradictions, the procedure described in the next section is called to identify and to remove the obsolete base belief set.

2.5. Removing Discovered Contradictions

The EBRS uses a diagnostic procedure to identify and to remove the obsolete beliefs. Precisely, the task of the diagnostic system is to return the set of *minimal candidates* of the obsolete base belief set, given the set of minimal inconsistent sets of base beliefs (called *minimal conflicts* following de Kleer and Williams (1987)). The set of minimal conflicts is exactly the collection of active environments in the labels of the contradiction nodes.

A candidate is a *hitting set* [3] of the set of minimal conflicts (Reiter, 1987). Thus, the problem of finding the set of minimal candidates is reduced to the problem of finding all minimal hitting sets of the set of minimal conflicts. The EBRS uses Reiter's *HS-tree* approach to solve the problem, but we introduce two new tree pruning strategies to improve efficiency of the algorithm.

Definition 1: An HS-tree for the given set family F is an edge-labeled and vertex-labeled tree T with the following properties:

(1) The root is labeled by Δ if F is empty. Otherwise, it is labeled by a set in F.

(2) For each vertex v_i of T, define $H(v_i)$ to be the set of edge labels on the path from the root to v_i. If v_i is labeled by Δ, then it has no descendant. If v_i is labeled by a set S_j in F, then for each element $a \in S_j$, v_i has a descendant vertex v_a connected with v_i by an edge labeled by a. The label for v_a is a set $S_k \in F$ such that $S_k \cap H(v_a) = \{\}$ if such an S_k exists. Otherwise, v_a is labeled by Δ.

(3) For each vertex v_i labeled Δ, $H(v_i)$ is a hitting set.

In Reiter's algorithm, vertices of the HS-tree are generated breadth-first. To reduce the size of the HS-tree, the algorithm uses the following two tree pruning strategies: [4]

(1) If v_i is a vertex labeled Δ, then any $v_j \neq v_i$ such that $H(v_i) \subseteq H(v_j)$ is not explored, since further exploration of v_j generates only supersets of a hitting set $H(v_i)$ which is already generated. A vertex not explored and not labeled Δ is labeled "X".

(2) If v_i is a vertex generated before v_j and $H(v_i) = H(v_j)$, then v_j is labeled X, since further exploration of v_i and exploration of v_j will generate two identical subtrees which contain the same set of hitting sets.

---

[3] Given a set family $F = \{S_i \mid i = 1, \ldots, n\}$, where each S_i is a set, a hitting set H for F is a set that contains at least one element of each set in F (see Garey and Johnson, 1979). For the diagnostic problem here, the set of minimal conflicts is a set family since each conflict is a set of base beliefs.

[4] There are actually three tree pruning strategies in Reiter's algorithm. The other one is irrelevant to our problem (see Huang, 1989).

We use the following new pruning strategies to further reduce the size of the HS-tree:

(3) Let v_i be the vertex being generated. If v_j is a vertex generated before v_i and labeled by a set in F, and if $H(v_j) \subseteq H(v_i)$, then v_i is labeled X.

(4) If v_i and v_j are siblings (having the same parent) and if the label of v_i has an element e_j identical to the label of the edge that connects v_j and their parent, then e_j should be removed from the label of v_i. However, if the label of v_j also contains e_i, the label of the edge connecting v_i and their parent, then remove only one element (e_j or e_i).

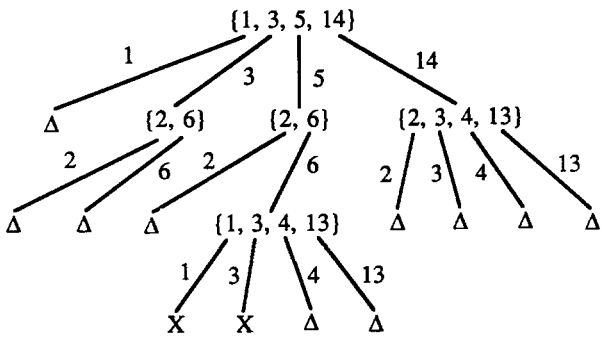

Fig. 2. The HS-Tree in the Example

THEOREM 1: Given a set family F, the HS-tree generated by the modified algorithm with pruning strategies (1)–(4) contains all minimal hitting sets of F which are exactly those $H(v_i)$'s such that v_i is labeled Δ.[5]

Strategy (2) in Reiter's algorithm is subsumed by strategy (3), so our algorithm actually uses only strategies (1), (3) and (4). The size of the HS-tree can be still reduced by pre-ordering the minimal conflicts so that smaller conflicts are put before larger conflicts, and that the conflicts having common elements are put together whenever it is possible. In this order, the vertices at the higher levels of the tree have fewer descendants, so fewer redundant subtrees are generated.

We summarize the algorithm of the diagnostic procedure below:

(1) Collect the active environments in the labels of the contradiction nodes to form the set of minimal conflicts.

(2) Pre-order the minimal conflicts in the way described above.

(3) Breadth-first generate the HS-tree of the set of minimal conflicts, using the three tree pruning strategies. The resulting set of minimal hitting sets is the set of minimal candidates.

[5] A proof of the theorem is provided in the Appendix.

(4) Select the obsolete base belief set from the minimal candidates (details are discussed below) and remove it from the system's environment.

Now we continue the example started in Section 2.4. The active environments in the labels of cont-1 and cont-2 are the minimal conflicts (step 1 of the algorithm). After pre-ordering (step 2), the set of minimal conflicts is:

$$\{\{1, 3, 5, 14\}, \{1, 2, 6, 14\}, \{1, 2, 3, 4, 13\}\}.$$

Using the three pruning strategies, the diagnostic procedure generates the HS-tree shown in Figure 2 (step 3). The resulting set of minimal candidates is:

$$\{\{1\}, \{2, 3\}, \{3, 6\}, \{2, 5\}, \{2, 14\}, \{3, 14\}, \{4, 14\}, \{13, 14\}, \{4, 5, 6\}, \{5, 6, 13\}\}.$$

Usually selection of the obsolete base belief set can be made from the minimal candidates if priorities are assigned to the beliefs. For example, in user/student modeling systems, normally we can assume that inference rules are more stable than factual beliefs (i.e., observed beliefs and derived beliefs) and thus assign a higher priority to the rules (see van Arragon (1990a) for use of priorities of beliefs in user modeling, and Fagin *et al.* (1983); Gardenfors and Makinson (1988) for priorities in general belief revision). Then the sets containing an inference rule are not considered.[6] In the example, only three minimal candidates are left:

$$\{\{1\}, \{2, 14\}, \{13, 14\}\}.$$

Furthermore, since new beliefs reflect the student's current knowledge state, they are assigned a higher priority than the old beliefs. Thus, the sets $\{2, 14\}$ and $\{13, 14\}$ can also be ruled out. The set $\{1\}$ is the only minimal candidate left and thus is selected as the obsolete base belief set. It is removed from the system's environment, which results in retraction of EBRS nodes 1, 8, 9, 10, 11 (i.e., propositions A, S, $\neg$T, U and V) from the belief set.[7] Now consider the updated Specific Deductive Beliefs (SDB). By excluding the inference rules of the belief set, SDB after revision is the beliefs corresponding to nodes 2, 12, 13 and 14, so

$$\text{SDB}(t_2) = \{\text{B, C, T, } \neg \text{ U}\}.$$

In some cases there may be several minimal candidates at the same prior-

[6] In fact, if we assume that inference rules are always true, we could even remove them earlier. We can remove them from each minimal conflict before the HS-tree algorithm is executed. Then the HS-tree would be smaller and generated faster.

[7] Recall that the EBRS removes only discovered contradictions, so there might be still some undiscovered contradictions in the updated belief set, although this is not the case in this example.

ity level (such cases would be rare if the priorities were designed carefully). In these cases, further measuring (e.g., directly questioning the student) may be necessary. A good discussion of the measurements and some techniques to design them can be found in de Kleer and Williams (1987).

3. Revision of Stereotypical Knowledge: A Revolutionary Process

3.1. The Default Package Network (DPN)

Although many stereotype structures have been used in user modeling systems, they don't seem to be suitable for modeling a student's changing knowledge. Grundy's stereotypes provide information about the users' personal traits (Rich, 1979). They group users by their social status (e.g., sex, age, occupation, etc.). At the beginning of each session Grundy asks the user for a self description. Stereotypes whose "triggers" match with the social status described by the user are then applied to the user. This approach may work well for modeling personal traits, but it does not carry over to modeling student knowledge. Students may not know how much they know in a domain they are learning. Even if they know, their measurement may be different from the system's. For example, a student's self description as a "novice" programmer likely does not coincide with the system's concept of a "novice" programmer.

Being aware of the unsuitability of using Grundy's approach to model knowledge, KNOME (Chin, 1989) does not ask the user for self description. It infers the user's knowledge level (the stereotype appropriate to the user) by looking for evidence that the user knows or doesn't know some key concepts. It collects this evidence during the first few interaction sessions with each particular user. (The evidence is akin to SMMS's observed beliefs, a part of deductive knowledge.) The evidence is matched with a number of pre-stored tables which indicate the user's likelihood of being at each knowledge level. However, once the user's knowledge level is determined, KNOME does not change it any more. This is not suitable for student modeling since a student's knowledge level changes constantly during the learning process. Also, KNOME has difficulty in dealing with stereotypes in related domains. This is more serious than it seems since usually a knowledge domain consists of several related sub-domains.

Stereotypes in GUMS are also used to model user knowledge (Finin, 1989). GUMS attempts the problem of revising stereotypical knowledge. In GUMS, user knowledge is organized in a stereotype tree where each node represents a class of users. The class inherits knowledge from all ancestor classes. The user model is revised when the application system observes new facts that conflict with the active stereotype. This is done by replacing the active stereotype with its closest ancestor that does not conflict with the observed facts. This treatment is often inappropriate, since revisions only

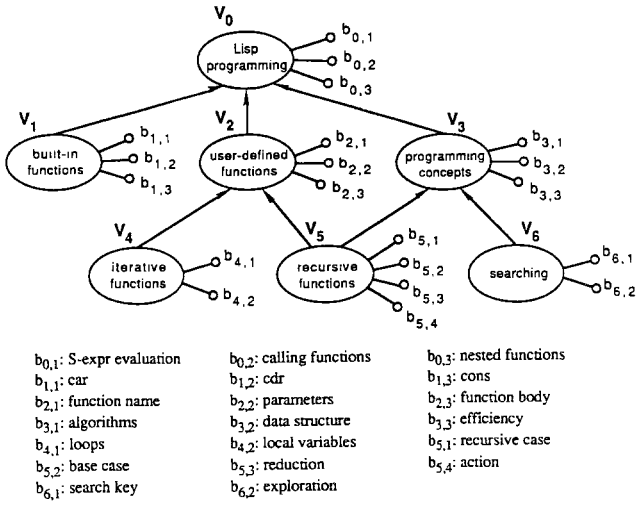

Fig. 3. A Fragment of a DPN for Lisp Programming Knowledge

decrease stereotypical knowledge about a user. After several interactions, even the root may conflict with the observed facts. Then the active stereotype becomes an empty set which is no longer useful. Also, GUMS allows only one active stereotype for each user. However, a user model often doesn't fit a single stereotype, but a combination of several stereotypes. For example, if the application system happened to know that the user is an "expert" in Unix and an "average" Lisp programmer, GUMS would have to give up information in one of these two stereotypes.

The stereotype structure in SMMS presented below is intended to overcome such difficulties in modeling knowledge. It handles relations between related stereotypes in different sub-domains and relations between stereotypical knowledge and deductive knowledge in a student model. In particular, it handles the dynamic properties of stereotypical knowledge in student modeling, providing an efficient revision algorithm for the knowledge which ensures that the revision activates and retains as many stereotypes as possible.

SMMS's stereotypical knowledge is represented in a directed acyclic graph called a *default package network (DPN)*. A DPN contains concepts and skills in a certain *domain*. The domain is divided into *sub-domains* each of which is further divided into smaller sub-domains. This forms a sub-domain hierarchy — the DPN in which each sub-domain is represented by a *node*, and the *general/specific* (super-domain/sub-domain) relations between the sub-domains are represented by the *links*. A link points from a specific node (a *child*) to a more general node (a *parent*). Figure 3 shows a segment of a DPN representing knowledge of Lisp programming.

Each sub-domain contains a subset of concepts (including misconcep-

tions) and skills. Each concept is described by a propositional formula (represented by a small circle in Figure 3) called a *d-proposition* (because stereotypical knowledge is used as defaults). In Figure 3, for example, the d-proposition *knows(S-expression-evaluation)* is labeled "S-expr evaluation" and denoted by $b_{0,1}$ (for it is the first d-proposition of V_0). Concepts in a sub-domain are divided into groups. Each group belongs to a child of the sub-domain, except for the group of the most general concepts in the sub-domain which belongs to the whole sub-domain. Thus, d-propositions in a DPN are partitioned into subsets. Each subset is attached to the node representing the most general sub-domain to which it belongs. For example, the d-proposition "knows(function-body)" (i.e, $b_{2,3}$) is attached to the "user-defined functions" node but not the "recursive functions" node.

Corresponding to an estimate of the tutoring system about a student's knowledge state in a sub-domain, a node in a DPN can be assigned a node *value* in a designated value range (e.g., ⟨NV, AV, EX⟩ for "novice", "average" and "expert"). A node value determines a package of d-propositions in the sub-domain which are assumed to be believed by the student at the corresponding knowledge level. Such a package is called a *d-proposition package* which is actually a local stereotype of the sub-domain). In particular, the d-proposition package corresponding to the value that is currently assigned to the student is called the *d-belief package* (the active stereotype of the sub-domain). d-propositions in the d-belief package are *d-beliefs*. For example, if we assign the value EX to the node V_3, then all three d-propositions of the node might be in the d-belief package. If we assign AV to V_3, however, then only $b_{3,2}$ and $b_{3,3}$ might be d-beliefs. Thus, the stereotypical beliefs (STB) of the student model are determined by the current value assignment of the nodes of the DPN. To account for the case that the system has no idea about the student's knowledge level in some sub-domains, a distinguished value "unknown" (denoted by UN) is defined. If a node has an UN value, then its d-belief package would be empty.

An estimate for a student's knowledge level in a specific sub-domain may be made according to the student's knowledge level in a more general sub-domain. Thus, the value of a node may determine the values of its children by default. In other words, the active stereotypes in a parent sub-domain may suggest the active stereotype for a child sub-domain. For example, if the student has an EX value in V_3, it might be reasonable to assign AV to V_5 and EX to V_6 (its two children) by default. These defaults reflect relations between stereotypes.

On the other hand, after a revision of deductive knowledge the *student model*, which is the union of the specific deductive beliefs (SDB) and the stereotypical beliefs (STB) with removal of each d-belief in the STB whose negation is a deductive belief in the SDB, may not be consistent with the active stereotype in a sub-domain any more. In this case, the active stereotype

in the sub-domain must be changed. Of course, revising deductive knowledge may not force the active stereotype to change if the conditions that the stereotypes remain active are not violated. These conditions are represented as *constraints* associated with the corresponding value of the node. The value may be assigned to the node only if the student model satisfies this set of constraints. These constraints represent relations between deductive knowledge and stereotypical knowledge in a student model. If the student model satisfies the constraint sets of several values of a node, then the value closest to the previous one should be chosen (so the value is changed only when necessary). The value of a node is not only constrained by the student model, but also by the values of its children. For example, we would not estimate a student to be an expert Lisp programmer if we believe that she/he has little knowledge of Lisp built-in functions. These constraints also reflect relations between stereotypes. Note that if a value of a node were flexible enough such that with this value any set of d-propositions could be believed and any value could be assigned to its children, then this value of the node would have no constraint.

Table I shows an example list of defaults and constraints corresponding to the DPN in Figure 3 (except those for V_1 and V_6 which are not relevant in the following discussion). We use the relations $\leq$, $=$ and $\geq$ to express constraints among nodes, based on a total ordering of the node values: NV < AV < EX. Also, we use SM, STB_i and $b_{i,j}$ to denote the student model, the d-belief package of V_i and the jth d-proposition of V_i, respectively. Designing the table of defaults and constraints seems to introduce more overhead to knowledge engineering of a DPN than other stereotype structures, but this may not be true. Other stereotype-based user modeling systems also must deal with relations between deductive knowledge (usually only observed information) and stereotypical knowledge and relations between stereotypes (Chin, 1989; Finin, 1989; Kobsa, 1990). The difference is that normally in these systems the constraints and defaults are represented implicitly in the procedures that deactivate and activate stereotypes, while in SMMS they are represented explicitly and manipulated by two simple and efficient constraint satisfaction and default propagation procedures, described in the next sub-section.

Also, since there is no overlapping d-propositions between nodes, designing constraints and defaults for a DPN is actually not difficult and quite flexible. In fact, only two conditions must be checked: (1) a default must not violate the constraints of the same value of the same node; (2) for each node, the defaults (and constraints) of a node must be weaker (containing fewer correct concepts and more misconceptions) than the values succeeding it in the total ordering of values. It is very easy to build an efficient program to automatically check the constraints and defaults for these two conditions.

TABLE I
Defaults and Constraints of the DPN in Figure 3

V_0:	EX:	defaults:	$STB_0=\{b_{0,1}, b_{0,2}, b_{0,3}\}$, V_1=AV, V_2=EX, V_3=EX;
		constraints:	$(\{b_{0,1}, b_{0,2}\} \subseteq SM) \vee (\{b_{0,1}, b_{0,3}\} \subseteq SM)$, $V_1 \geq AV$, V_2=EX, $V_3 \geq AV$;
	AV:	defaults:	$STB_0=\{b_{0,1}, b_{0,2}\}$, V_1=AV, V_2=EX, V_3=AV;
		constraints:	$\{b_{0,1}\} \subseteq SM$, $V_2 \geq AV$, $V_3 \leq AV$;
	NV:	defaults:	$STB_0=\{b_{0,1}\}$, V_1=UN, V_2=NV, V_3=AV;
		constraints:	$\{b_{0,3}\} \not\subseteq SM$, $V_2 \leq AV$, V_3=NV;
V_2:	EX:	defaults:	$STB_2=\{b_{2,1}, b_{2,2}\}$, V_4=EX, V_5=EX;
		constraints:	$\{b_{2,1}, b_{2,2}\} \subseteq SM$, $V_4 \geq AV$, $V_5 \geq AV$;
	AV:	defaults:	$STB_2=\{b_{2,1}, b_{2,2}\}$, V_4=AV, V_5=AV;
		constraints:	$(\{b_{2,1}\} \subseteq SM) \wedge (\{b_{2,3}\} \not\subseteq SM$, $V_4 \leq AV$, $V_5 \leq AV$;
	NV:	defaults:	$STB_2=\{\}$, V_4=NV, V_5=NV;
		constraints:	$(\{b_{2,2}\} \not\subseteq SM) \wedge (\{b_{2,3}\} \not\subseteq SM)$, V_4=NV, $V_5 \leq AV$;
V_3:	EX:	defaults:	$STB_3=\{b_{3,1}, b_{3,2}, b_{3,3}\}$, V_5=EX, V_6=EX;
		constraints:	$\{b_{3,2}, b_{3,3}\} \subseteq SM$, $V_5 \geq AV$, V_6=EX;
	AV:	defaults:	$STB_3=\{b_{3,2}, b_{3,3}\}$, V_5=AV, V_6=EX;
		constraints:	$(\{b_{3,1}\} \not\subseteq SM) \wedge (\{b_{3,2}\} \subseteq SM)$, $V_6 \geq AV$;
	NV:	defaults:	$STB_3=\{\}$, V_5=NV, V_6=AV;
		constraints:	$(\{b_{3,1}\} \not\subseteq SM) \wedge (\{b_{3,2}\} \not\subseteq SM)$, $V_5 \leq AV$, $V_6 \leq AV$;
V_4:	EX:	defaults:	$STB_4=\{b_{4,1}, b_{4,2}\}$;
		constraints:	$\{b_{4,1}\} \subseteq SM$;
	AV:	defaults:	$STB_4=\{b_{4,1}\}$;
		constraints:	$(\{b_{4,1}\} \subseteq SM) \wedge (\{b_{4,2}\} \not\subseteq SM)$;
	NV:	defaults:	$STB_4=\{\}$;
		constraints:	$(\{b_{4,1}\} \not\subseteq SM) \wedge (\{b_{4,2}\} \not\subseteq SM)$;
V_5:	EX:	defaults:	$STB_5=\{b_{5,1}, b_{5,2}, b_{5,4}\}$;
		constraints:	$\{b_{5,2}, b_{5,4}\} \subseteq SM$;
	AV:	defaults:	$STB_5=\{b_{5,1}, b_{5,2}\}$;
		constraints:	$(\{b_{5,2}\} \subseteq SM) \wedge (\{b_{5,3}\} \not\subseteq SM)$;
	NV:	defaults:	$STB_5=\{b_{5,2}\}$;
		constraints:	$(\{b_{5,1}\} \not\subseteq SM) \wedge (\{b_{5,3}\} \not\subseteq SM)$.

3.2. Constraint Satisfaction and Default Propagation

A revision of stereotypical knowledge occurs when a revision of deductive knowledge violates a constraint of the value of a DPN node, namely the activation conditions of an active stereotype. In this case, the value of the node must be changed (i.e., the active stereotype of the sub-domain must be changed). After the change, however, the node's new value may violate a constraint of the value of its parents. Thus, value change may propagate upwards. This process is called *constraint satisfaction.* For example, using Figure 3 and Table I, assume V_0 = EX, V_3 = AV, V_6 = AV. If V_6's value is forced to change to NV, then a constraint of V_3's AV value would be violated.

Thus, V_3 must be changed to NV as well, which in turn would force V_0 to change to AV. On the other hand, the value of a node may determine the values of its children by default, so changing the value of a node may cause the value of a child to be changed if the child's current value is "unknown" or was determined by a default of the previous value of the node. For example, if V_4's AV value was determined by V_2's previous value AV and now V_2's value is changed to EX, then V_4's value would be changed to EX because of the default assignment of V_2's new EX value. Thus, value assignment in a DPN may also propagate downwards. This process is called *default propagation.* Note that when a value resulting from constraint satisfaction conflicts with a value resulting from default propagation, the result of constraint satisfaction has a higher priority since it comes from a more concrete information source (i.e., deductive knowledge).

In general, an evolutionary revision of a student model that occurs in its deductive knowledge may force changes of active stereotypes in some sub-domains. This is a *local revolutionary revision* of the student model. In addition, changing active stereotypes in these sub-domains may trigger a bottom-up constraint satisfaction process, followed by a top-down default propagation process, changing active stereotypes in many other sub-domains. Then, the stereotypical knowledge base (and thus also the whole student model) undergoes a *global revolutionary revision.* The ratio of revolutionary revisions to evolutionary revisions depends on the tolerance of the constraints designed for the DPN.

Even if the constraints are designed carefully, there might be cases in which the student model doesn't satisfy the constraint set of any value of a node. This usually happens when a node, say V_i, is forced to change value during constraint satisfaction, while the new value to be assigned also has some constraint not satisfied (called a *second violation*). If the second violation comes from a child whose current value was determined by the previous value of V_i, then the second violation would be removed in the next default propagation, and thus ignored, so the new value is still assigned. Otherwise, V_i is assigned the value "unknown". Here "unknown" means "unclassifiable", which may be slightly different from its original intuition "having no idea", but the same semantics applies. An important property of the "unknown" value is that it is a "wild card" value which can satisfy any constraints and that itself has no constraint. Thus, the value of a node is not affected by its parents or its children with an "unknown" value. This has the advantage that failure of the system can be restricted to the local level, namely a single sub-domain, similar to what has been achieved in using a granularity hierarchy for recognition (Greer and McCalla, 1989).

Another use of the "unknown" value is to avoid circularity. Assume that the value of a node V_a is revised in the constraint satisfaction process. Then in the default propagation process, a child V_c of V_a with the "unknown"

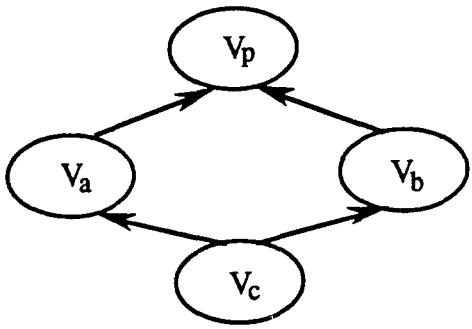

Fig. 4. A Revision Circle in a DPN

value might be set to a new value. But if this new value violates the constraints of a parent, V_b, of V_c, then V_c's value would remain "unknown". This avoids another constraint satisfaction process which might cause a revision circle shown in Figure 4.

3.3. The Algorithm for Revision of Stereotypical Knowledge

We now summarize the algorithm for revision of stereotypical knowledge in a student model. To clarify how a revision of deductive knowledge triggers a revision of stereotypical knowledge, we present the whole algorithm for the SMMS, but abstract the part for the revision of deductive knowledge into the first two lines (recall that SDB is the set of specific deductive beliefs and STB is the set of stereotypical beliefs):

Algorithm SMMS:
- Compute the updated SDB in response to new information (using EBRS)
- Compute the student model (from the updated SDB and the current STB)
- Satisfy the constraints in the DPN (bottom-up)
- Propagate value changes along with the DPN according to the defaults (top-down)
- If any change in the DPN is made, then
  * Compute the updated STB (according to the updated DPN)
  * Compute the student model again (from the updated SDB and the updated STB)

End. {of the SMMS algorithm}

Each step of the algorithm is a procedure. The procedure for updating SDB (i.e., the EBRS algorithm) has been discussed in Sections 2.3–2.5. Since an ATMS and a diagnostic system are used, the EBRS requires time

exponential in the size of the knowledge base in the worst case. STB is the union of the d-belief packages of the nodes in the DPN. The union operation costs at most O(M log M) time, where M is the size of the DPN. The student model is the union of SDB and STB after removing every stereotypical belief in the STB which directly contradicts a deductive belief in SDB (i.e., if p is in STB while $\neg p$ is in SDB, then p is removed). By sorting STB and SDB, the operations of union and contradiction removal (the computation of the new student model) can be done in O(N log N) time, where N = L + M, and L is the size of SDB (Sedgewick, 1988).

Now we present the two procedures that update the DPN. In the procedures, we assume that the part of student model related to a node V_i of the DPN is accessed by the function SM(i). Each node has a flag "dp". If the current value of V_i was determined by the default of one of its parents, say V_p, then $dp(V_i) = V_p$. Otherwise, $dp(V_i) = 0$. Also, the function children(i) returns the set of values of V_i's children. The function child(i, j) returns the value of V_i's jth child. We further assume the nodes in the DPN to be numbered in the order of top (i.e., the root) to bottom, and at the same level from left to right, as is shown in Figure 3.

Procedure Constraint-Satisfaction (bottom-up)

For i := N down to 0, do
  If neither SM(i) nor children(i) is changed, or if the changes do not violate any constraint of the value of V_i, then
  do nothing
else
  - Compute the new value of V_i according to new SM(i) and new children(i)
  - If a constraint of the new value of V_i is violated and the violation is not from a child V_c such that $dp(V_c) = V_i$, then
      $V_i \leftarrow$ UN
  - $dp(V_i) \leftarrow 0$
End; {constraint-satisfaction}

Procedure Default-Propagation (top-down)

For i:= 0 to N, do
  If the value of V_i is not changed, then
      do nothing
  else for each child of V_i (note: child(i, j) is the one being dealt with), do
      If child(i, j) = UN or dp(child(i, j)) = V_i, then
      - Assign a new value to child(i, j) according to the default of the new value of V_i
      - $dp(child(i, j)) \leftarrow V_i$
      - Check whether child(i, j) violates a constraint of the value of each of child(i, j)'s parents (except V_i) or not
      - If a violation occurs, then
        - child(i, j) $\leftarrow$ UN
        - $dp(child(i, j)) \leftarrow 0$
End; {default-propagation}

Both procedures use computational time linear in the size of the DPN, if the degree of each node (the number of links associated with the node) is limited by some constant which does not depend on the size of the DPN. This is the case with most student modeling and user modeling systems. Thus, excluding the first step at which the EBRS is executed, the rest of the SMMS algorithm requires time O(N log N). If the condition that the degree of each node is not greater than a constant is not true, then each procedure above requires time $O(M^2)$. The SMMS algorithm (excluding the first step) requires time $O(M^2 + L \log L)$. This analytical result tells that if a more efficient algorithm for revising deductive knowledge is developed [8], then the SMMS algorithm would be very efficient, using the current DPN for revision of stereotypical knowledge.

3.4. An Example

Here we give an example of revising stereotypical knowledge in a student model. The revision is triggered by an evolutionary revision of the deductive knowledge. The example uses the DPN displayed in Figure 3 and its defaults and constraints defined in Table I. We first assume that at time t_0, before the revision happens, the set of deductive beliefs SDB is

$$SDB(t_0) = \{b_{0,1}, \neg\, b_{2,1}, b_{3,1}, b_{5,2}\},$$

and the value assignment to the DPN at time t_0 is:

$$V_0 = AV, V_1 = UN, V_2 = AV, V_3 = NV,$$

$$V_4 = UN, V_5 = AV, V_6 = UN.$$

The set of stereotypical beliefs STB is the union of the d-belief packages of the nodes determined by the defaults of this value assignment. Thus, we have

$$STB(t_0) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{5,1}, b_{5,2}\}.$$

The student model is the union of SDB and STB with removal of each d-belief in STB that directly contradicts a deductive belief in SDB. Therefore,

$$SM(t_0) = \{b_{0,1}, b_{0,2}, \neg b_{2,1}, b_{2,2}, b_{3,1}, b_{5,1}, b_{5,2}\}.$$

At time t_1, assume that three new deductive beliefs, $\neg b_{5,1}$, $b_{5,3}$ and $b_{5,4}$, are obtained, and that EBRS removes a deductive belief $\neg b_{2,1}$ to maintain consistency of the deductive knowledge base. Then SDB becomes:

$$SDB(t_1) = \{b_{0,1}, b_{3,1}, \neg b_{5,1}, b_{5,2}, b_{5,3}, b_{5,4}\}.$$

[8] Such an efficient algorithm has been recently developed in (Huang et al., 1991).

The stereotypical beliefs are not yet revised at t_1. Thus, $STB(t_1) = STB(t_0)$. The student model at t_1 is:

$$SM(t_1) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{3,1}, \neg b_{5,1}, b_{5,2}, b_{5,3}, b_{5,4}\}.$$

This shows an evolutionary revision of the student model.

Now, revision of stereotypical beliefs starts. By checking Table I, one can find that a constraint for assigning AV to V_5 is violated since SM contains $b_{5,3}$ now. Thus, V_5 is upgraded and assigned an EX value. Constraint satisfaction propagates from V_5 to V_2 and V_3. The value of V_2 changes from AV to EX as well, since a constraint for V_2 to keep its AV value, "$V_5 \leq$ AV", is violated. For V_3, a constraint of its NV value, "$V_5 \leq$ AV", is also violated. However, an AV value cannot be assigned to it (nor can an EX value, of course), since SM does not contain $b_{3,2}$, which also violates a constraint for V_3 to have an AV value. Thus, V_3 is assigned UN, which means that the system cannot classify the student's knowledge level in this sub-domain. Although two children of V_0 have their values changed, no constraint for its AV value is violated. Thus, V_0 keeps the value unchanged.

Then the default propagation procedure is executed. Since V_2's first child V_4 has an UN value at this time, it is assigned an EX value according to a default of V_2's EX value. Thus, at time t_2 when revision is completed, the value assignment to the DPN is

$$V_0 = \text{AV}, V_1 = \text{UN}, V_2 = \text{EX}, V_3 = \text{UN},$$
$$V_4 = \text{EX}, V_5 = \text{EX}, V_6 = \text{UN}.$$

By taking the union of corresponding d-belief packages of the nodes,

$$STB(t_2) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{4,1}, b_{4,2}, b_{5,1}, b_{5,2}, b_{5,4}\}.$$

Finally, since $SDB(t_2) = SDB(t_1)$, the updated student model is

$$SM(t_2) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{3,1}, b_{4,1}, b_{4,2}, \neg b_{5,1}, b_{5,2}, b_{5,3}, b_{5,4}\}.$$

Thus, by changing values of V_2, V_3, V_4 and V_5 (active stereotypes in the corresponding sub-domains), a drastic change, or a revolutionary revision, has occurred in the student model.

4. Comparison with Related Work

This section compares the SMMS with related work in four aspects of user/student modeling: (1) handling deductive knowledge; (2) activation and deactivation of stereotypes; (3) stereotypes in related knowledge domains; (4) conflicts between deductive knowledge and stereotypical knowledge. Since most student modeling systems developed by ITS researchers do not deal with inferences over the existing student model and revision of the student model formed by such inferences, most of the related work discussed

here is from the area of general user modeling. In particular, we consider four important user modeling systems: Grundy (Rich, 1979, 1989), KNOME (Chin, 1989), GUMS (Finin and Drager, 1986; Finin, 1989) and BGP-MS (Kobsa, 1990).

(1) *Handling deductive knowledge*
Many user modeling systems restrict their deductive knowledge to observed information (Rich, 1989; Chin, 1989). They do not make inferences to augment deductive knowledge. But this kind of inference is especially important for a student modeling system since concepts in a subject that a student is learning are usually more structural (Goldstein, 1979; Sleeman, 1985). Knowing a concept usually implies knowing some other concepts and not holding certain related misconceptions. Similarly, believing a misconception may be evidence of believing other misconceptions. GUMS's deductive inference rules are built only inside stereotypes. This may allow better control of the inference rules in that they are applied to only certain classes of users. However, most of these inference rules seem to be stereotype-independent (Kass and Finin, 1987; Kass, 1990; Sleeman, 1985). Installing rules inside stereotypes may unnecessarily increase complexity of the stereotype structure and create duplicate inference rules. BGP-MS's approach is similar to ours except that we provide only a framework that allows installing the inference rules, while BGP-MS actually builds a set of rules in the system.

None of these systems, except GUMS, deals with *revision* of deductive knowledge. GUMS accomplishes revision by using observed facts to override defaults. This approach is studied more extensively by van Arragon (1990a, 1990b). However, the approach does not always work. In particular, it is inappropriate for applications such as intelligent tutoring in which a user's knowledge state may change during interactions. For example, if yesterday the tutor knew very surely that the student did not know how to login to the computer, then $\neg$ *knows(login)* is a fact, entered yesterday, in the student model. However, if today the tutor observes that the student has learned how to login, then *knows(login)* is also a fact, entered today, in the student model. Which fact should override the other? Obviously, a simple solution for this situation is to remove the obsolete belief $\neg$ *knows(login)*, as the EBRS does, rather than an overriding scheme provided by a default reasoning system.

(2) *Activation and deactivation of stereotypes*
How to activate and deactivate stereotypes is a central problem in many stereotype-based systems. We have given a related discussion of this problem in Section 3.1, so duplicate material will be presented briefly in the following discussion. Grundy relies on the user's input about her/his social status to activate stereotypes. An active stereotype is not retracted during

interaction. This approach may work for modeling personal traits, but it is not suitable for modeling knowledge. KNOME's approach is closer to ours in that it uses deductive knowledge (only observed information) to determine active stereotypes, but still it does not retract activated stereotypes. GUMS classifies beliefs in a stereotype into two groups: *definite beliefs* and *default beliefs.* If the observed information conflicts with a definite belief in the active stereotype, then the stereotype is deactivated. This approach is somewhat similar to our use of constraints. However, deactivating and activating stereotypes in GUMS is done using an overly simple method. The deactivated stereotype is trivially replaced by a more general stereotype that has no definite belief conflicting with the observed information. Thus, stereotypical knowledge in a user model constantly decreases. BGP-MS provides a set of functions for specifying activation and retraction conditions of each stereotype in terms of deductive beliefs in the user model. A stereotype would be activated if its activation conditions are satisfied and would be deactivated if its retraction conditions are satisfied. This set of pre-defined functions is very similar to SMMS's constraint language except that the constraint language is more general and deals also with relations between stereotypes. In addition, activation and deactivation in the SMMS is accomplished by two well established and efficient procedures: constraint satisfaction and default propagation.

(3) *Stereotypes in related knowledge domains*
We must deal with the problem of how to activate stereotypes in related domains if we model the user's domain knowledge. This problem is not addressed by Grundy and BGP-MS. KNOME recognizes the problem but does not deal with it (Chin, 1989, p. 106). GUMS allows activating stereotype in only one domain. If a user is identified as a "Unix Hacker", for example, then GUMS is unable to use knowledge in stereotypes in other domains, such as "Lisp Machine Expert", that is also applicable to the user. The SMMS offers a better solution to this problem. Stereotypes in different domains may be activated simultaneously based on deductive knowledge and active stereotypes in related domains. A set of constraints is used to ensure consistency of active stereotypes.

(4) *Conflicts between deductive knowledge and stereotypical knowledge*
How to resolve conflicts between deductive knowledge and stereotypical knowledge is also a research issue in user modeling. Except for KNOME, all other systems cope with the problem, although deductive knowledge in some of these systems contains only observed beliefs. The three systems agree that a deductive belief would override a stereotypical belief if they are in conflict. The discrepancy is in whether such a conflict can trigger deactivation of a stereotype. Grundy never retracts an activated stereotype

in response to a conflict, while GUMS and BGP-MS might deactivate some stereotypes if the conflict is serious enough to violate some conditions for keeping the stereotypes active. The latter approach is also used by SMMS. The difference among the three systems (GUMS, BGP-MS and SMMS) is in their representations of these conditions, as we mentioned during our discussion of issue (2) earlier in this section.

5. Conclusions

We have investigated the issue of revising deductive knowledge and stereotypical knowledge in a student model and shown how to use reason maintenance and formal diagnosis techniques to accomplish revision of the deductive knowledge base. The DPN has been developed to account for the tutor's stereotypical knowledge. An efficient algorithm that satisfies constraints and propagate defaults of the DPN is designed for revising the stereotypical knowledge base. We have also shown how a revision of deductive knowledge triggers a revision of stereotypical knowledge, resulting in a desirable updated student model in which two types of knowledge exist harmoniously. The student model maintenance system (SMMS) described in the paper is domain independent. Its research results apply to a variety of student models and user models.

Appendix: Proof of Correctness of the HS-Tree Algorithm

Here we prove that the modified HS-tree algorithm presented in Section 2.5 correctly returns the set of all minimal hitting sets of the given set family F, providing that each set in F is a minimal conflict (Theorem 1). In other words, we prove that for a vertex v_i, $H(v_i)$ is a minimal hitting set of F if and only if v_i is labeled by Δ.

Proof: we first prove correctness of Reiter's algorithm with pruning strategies (1) and (2).

LEMMA 1: A minimal hitting set of F must equal $H(v_i)$ for some v_i labeled by Δ in the HS-tree generated by Reiter's algorithm with pruning strategies (1) and (2).

Proof: First, if the algorithm uses no pruning strategy, Lemma 1 is trivially true by construction of the HS-tree. Pruning strategy (1) prunes only supersets of generated minimal hitting sets, and strategy (2) removes only duplicate branches, so the two strategies remove no minimal hitting set. □

LEMMA 2: In an HS-tree generated by Reiter's algorithm with pruning strategies (1) and (2), for every vertex v_i labeled Δ, $H(v_i)$ is a minimal hitting set.

Proof: First, $H(v_i)$ is a hitting set by definition. Second, $H(v_i)$ is not a superset of $H(v_j)$ for any previously generated vertex v_j labeled Δ, for otherwise v_i would be labeled X. Third, because the HS-tree is generated breadth-first, no $H(v_k)$ is smaller than $H(v_i)$ for any subsequently generated vertex v_k labeled Δ, so $H(v_i)$ is not a proper superset of $H(v_k)$. Note that all minimal hitting sets are contained in the resulting HS-tree (Lemma 1). Thus, $H(v_i)$ is minimal. □

Now we show that the other two pruning strategies, (3) and (4) used in the modified HS-tree algorithm, preserve the results of Lemma 1–2. The result of Lemma 2

is obviously preserved since since by applying pruning strategies, no new vertex labeled Δ can be generated. Thus, we need only to show the result of Lemma 1 for the modified algorithm, that is, to show that pruning strategies (3) and (4) preserve every minimal hitting set.

LEMMA 3: Pruning strategy (3) preserves every minimal hitting set.

Proof: With no pruning strategy (3), whenever $H(v_j)$ is not a hitting set but is a subset of some minimal hitting set, the leaves of the subtree rooted at v_j include every v_l such that $H(v_l)$ is a minimal hitting set containing $H(v_j)$. But then v_j also generates all minimal hitting sets containing $H(v_i)$, if any, since $H(v_j) \subset H(v_i)$. Thus, there is no harm in pruning v_i. The other cases are trivial: if $H(v_j)$ is a hitting set, then expanding $H(v_i)$ is a waste; if $H(v_j)$ is not a subset of any minimal hitting set, then $H(v_i)$ yields no minimal hitting set, and no harm is done pruning v_i. □

LEMMA 4: Pruning strategy (4) preserves every minimal hitting set.

Proof: Let v_i and v_j be siblings connected to parent v_p by edges labeled e_i and e_j, respectively. Note that the algorithm does not depend on the order that the vertices generated, so suppose v_i is generated before v_j, and its label contains e_j. Let v_c be the child of v_i that would be connected to v_i by the edge labeled e_j. Then $H(v_c) = H(v_p) \cup \{e_i, e_j\}$ and $H(v_j) = H(v_p) \cup \{e_j\}$, so $H(v_j) \subset H(v_c)$. By Lemma 3, v_c can be pruned. Equally, we can remove e_j from the label of v_i. On the other hand, suppose v_j is generated before v_i, a similar argument applies. Note that we obviously cannot remove both. □

By combining Lemma 1–4, we have completed the proof of Theorem 1. □

Acknowledgements

Thanks to the University of Saskatchewan and the Natural Sciences and Engineering Research Council for their financial support. Also thanks to the reviewers of this paper for their valuable comments and suggestions.

References

Alchourron, C. and D. Makinson: 1982, 'On the Logic of Theory Change: Contraction Functions and Their Associated Revision Functions'. *Theoria* **48**, 14–37.

Alchourron, C., P. Gardenfors, and D. Makinson: 1985, 'On the Logic of Theory Change: Partial Meeting Contraction and Revision Functions. *Journal of Symbolic Logic* **50**(2), 510–530.

Burton, R. R. and J. S. Brown: 1982, 'An Investigation of Computer Coaching for Informal Learning Activities'. In: D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*, Harcourt Brace Jovanovich, pp. 79–98.

Chin, D. N.: 1989, 'KNOME: Modeling What the User Knows in UC'. In: A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, pp. 74–107.

Clancey, W. J.: 1986, 'Qualitative Student Models'. In: J. F. Traub (ed.), *Annual Review of Computer Science 1*, pp. 381–450.

Clancey, W. J.: 1987, *Knowledge-Based Tutoring: The GUIDON Program*, The MIT Press.

Dalal, M.: 1988, *Investigation into a Theory of Knowledge Base Revision: Preliminary Report.* Proceedings AAAI-88, Saint Paul, MN, pp. 475–479.

de Kleer, J.: 1986, 'An Assumption-Based TMS'. *Artificial Intelligence* **28**(2), 127–162.

de Kleer, J. and B. C. Williams: 1987, 'Diagnosing Multiple Faults'. *Artificial Intelligence* **32**, 97–130.

Doyle, J.: 1979, 'A Truth Maintenance System'. *Artificial Intelligence* **12**, 231–272.

Fagin, R., J. D. Ullman, and M. Y. Vardi: 1983, 'On the Semantics of Updates in Databases'. *Proceedings of the Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Atlanta, pp. 352–365.

Finin, T.: 1989, 'GUMS — A General User Modeling Shell'. In: A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, pp. 411–430.

Finin, T. and D. Drager: 1986, *GUMS1: A General User Modelling System.* Proceedings CSCSI-86, Montreal, Canada, pp. 24–30.

Gardenfors, P.: 1984, 'Epistemic Importance and Minimal Changes of Belief'. *Australasian Journal of Philosophy* **62**(2), 136–157.

Gardenfors, P.: 1990, 'The Dynamics of Belief Systems: Foundations vs. Coherence Theories'. *Revue Internationale de Philosophie*, to appear.

Gardenfors, P. and D. Makinson: 1988, 'Revision of Knowledge Systems Using Epistemic Entrenchment'. In: M. Y. Vardi (ed.), *Proceedings of the Second Conferences on Theoretical Aspects of Reasoning about Knowledge*, Morgan Kaufmann Publishers,Inc., pp. 83–95.

Garey, M. R. and D. S. Johnson: 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company.

Goldstein, I. P.: 1979, 'The Genetic Graph: A Representation for the Evolution of Procedural Knowledge'. *Int. J. Man-Machine Studies* **11**, 51–77.

Greer, J. E. and G. I. McCalla: 1989, *A Computational Framework for Granularity and Its Application to Educational Diagnosis.* Proceedings IJCAI-89, Detroit, Michigan, pp. 477–482.

Harman, G.: 1986, *Change in View: Principles of Reasoning.* MIT Press, Cambridge, Massachusetts.

Huang, X.: 1989, 'A Study of the Hitting Set Problem'. Manuscript, Department of Computational Science, University of Saskatchewan.

Huang, X., G. I. McCalla and E. Neufeld: 1991, 'Using Attention in Belief Revision'. proceedings AAAI-91, Anaheim, California (to appear).

Kass, R.: 1990, 'Building a User Model Implicitly from a Cooperative Advisory Dialog'. *Advance Papers of the 2nd International Workshop on User Modeling*, Honolulu, HI.

Kass, R. and T. Finin: 1987, *Rules for the Implicit Acquisition of Knowledge about the User.* Proceedings AAAI-87, Seattle, pp. 295–300.

Kimball, R.: 1982, 'A Self-Improving Tutor for Symbolic Integration'. In: D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems Harcourt Brace Jovanovich.*

Kobsa, A.: 1990, 'Modeling the User's Conceptual Knowledge in BGP-MS, a User Modeling Shell System'. *Computational Intelligence* **6**(4).

Makinson, D.: 1985, 'How to Give It Up: A Survey of Some Formal Aspects of the Logic of Theory Change'. *Synthese* **62**, 347–363.

Martins, J. P. and S. C. Shapiro: 1988, 'A Model for Belief Revision'. *Artificial Intelligence* **35**(1), 25–79.

McCalla, G. I., J. E. Greer, and the SCENT Research Team: 1988, *Intelligent Advising in Problem Solving Domains: The SCENT-3 Architecture.* Proceedings ITS-88, Montreal, pp. 124–131.

Reiter, R.: 1980, 'A Logic for Default Reasoning'. *Artificial Intelligence* **13**, 81–132.

Reiter, R.: 1987, 'The Theory of Diagnosis from First Principles'. *Artificial Intelligence* **32**(1), 57–95.

Rich, E.: 1979, 'User Modelling via Stereotypes'. *Cognitive Science* **3**, 329–354.

Rich, E.: 1989, 'Stereotypes and User Modeling'. In: A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, pp. 35–51.

Ross, L. and C. A. Anderson: 1982, 'Shortcomings in the Attribution Process: on the Origins and Maintenance of Erroneous Social Assessments'. In: D. Kahneman, P. Slovic, and A. Tversky (eds.), *Judgement Under Uncertainty: Heuristics and Biases*, Cambridge University Press, Cambridge, pp. 129–152.

Sedgewick, R.: 1988, *Algorithm*. Addison-Wesley Publishing Company.
Sleeman, D.: 1985, 'UMFE: A User Modelling Front-End Subsystem'. *International Journal of Man-Machine Studies* **23**, 71–88.
Sleeman, D. and J. S. Brown (eds.): 1982, *Intelligent Tutoring Systems*, Harcourt Brace Jovanovich.
van Arragon, P.: 1990a, *Nested Default Reasoning with Priority Levels*. Proceedings CSCSI-90, Ottawa, pp. 77–83.
van Arragon, P.: 1990b, *Nested Default Reasoning for User Modeling*. Research Report CS-90-25, Department of Computer Science, University of Waterloo.
Wahlster, W. and A. Kobsa: 1989, 'User Models in Dialog Systems'. In: A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, pp. 5–34.
Wenger, E.: 1987, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Inc., Los Altos, CA.