Nonmonotonic Model Inference
— *A Formalization of Student Modeling* —

Mitsuru IKEDA, Yasuyuki KONO and Riichiro MIZOGUCHI
I.S.I.R., Osaka University
8-1, Mihogaoka, Ibaraki, 567, Japan
{ikeda,kono,miz}@ei.sanken.osaka-u.ac.jp

Abstract

A student model description language and its synthesis method are presented. The language called SMDL is based on a logic programming language taking 4 truth values such as true, false, unknown and fail. A modeling method called HSMIS is a new nonmonotonic model inference system and has the following major characteristics: (1) Model inference of logic program taking 4 truth values, (2)Treatment of nonmonotonicity of both student's belief and inference process itself. HSMIS incorporates de Kleer's ATMS as a vehicle for formulating the nonmonotonicity. Both SMDL interpreter and HSMIS have been implemented in Common ESP(Extended Self-contained Prolog) and incorporated into a framework for ITS, called FITS.

1 Introduction

Student modeling is one of the most important topics of ITS research, because the behavior of an ITS largely depends on a student model, which represents the snapshot of student's knowledge. This is a reason why many efforts concerning student modeling have been made, for instance, overlay model, buggy model, perturbation model, etc.[Wenger, 1987]. Most of the conventional modeling methods have simple pragmatic structures and have been incorporated into many ITS's. However, all the methods have some limitations and no complete and sound inference procedure for the models is obtained yet. In this paper, we formalize a student modeling problem as an inductive inference problem, that is, a problem to construct a model explaining observed data. In our case, data are student's answers and the model is student's knowledge.

In order to make ITS's intelligent, student models have to satisfy the following requirements:

1. Accuracy-cost tradeoff: In general, the more accurate the student model becomes, the more effective the behavior of the system becomes. However, there exists trade-off between accuracy of the model and cost to construct it. From a pragmatic viewpoint, we must set up an appropriate representation scheme for student models by taking the trade-off into considerations.

2. Nonmonotonicity: Tutoring is to guide students toward better understanding of teaching material. This means that the learning process is essentially attained with change of their minds and hence the consistency of student's answers can be easily lost. Therefore, student modeling methods should be able to automatically manage the consistency of student's answers in order to follow the student's mind. However, there is very few attempts to formulate the nonmonotonicity of student modeling process[Burton, 1982][Huang *et al.*, 1991a].

3. Unknown assertions: When a student fails to deduce her own solution for a problem, she would say to her teacher "I could not solve the problem". Needless to say, this assertion does not mean she does not have any knowledge. The student model module should use this assertion as informative data about her knowledge and construct a model which explains why she cannot deduce the answer from her own knowledge. This requires student model to deduce "unknown" assertions.

4. Theoretical foundation: Domain-independent and theoretical foundation for the student modeling mechanism should be defined. It contributes to both clarification of the inherent property of student modeling problem and to articulation of the scalability and reusability of the proposed mechanism.

To meet these requirements, we have developed a student model description language SMDL and a hypothetical student model inference system HSMIS. SMDL is an extended version of Prolog and takes four truth values including "unknown" to model the student precisely. HSMIS, an extended version of Shapiro's MIS[Shapiro, 1982], is an inductive inference system for SMDL. The second requirement mentioned above suggests that the inference procedure should cope with nonmonotonic modeling process. In HSMIS, ATMS: Assumption-based Truth Maintenance System [de Kleer, 1986] is employed for this purpose. HSMIS has been implemented in Common ESP(Extended Self-contained Prolog) on SPARC station.

2 SMDL : A Student Model Description Language

In addition to the above requirements, a student model is required to represent not only students but also sys-

tems' understanding of the students, which implies the model has to distinguish the two states: The system can predict the behavior of the student and the system cannot. When the model is based on logic, which is our case, it has to have two truth values, true and false, which denote the above two states, respectively. Needless to say, the former state, that is, one corresponding to "true", should represent the student's logical state such as "true", "false", and "unknown" which stand for "the student believes a statement is true", "the student believes it is false" and "the student does not ascertain its truth", respectively. Then, we have two seemingly same truth values "false", which can be discriminated as follows: Employing Prolog terminology, the former "false" is treated as "fail" and the other as one of the three values corresponding to "success". Discrimination among the three values is done by introducing an auxiliary argument interpreted by meta-interpreter.

2.1 Overview of SMDL

Facts are represented in SMDL as follows.

```
temperate(paris,true).  torrid(paris,false).
fertile(paris,unknown).
```

These three facts represent "the student believes Paris is not in the torrid zone but in the temperate zone and does not know whether it is fertile or not." If the facts are all in the student model, it also represents "the system cannot say anything about student's knowledge for other statements."

Clauses are written in the form of

$$A :: -B_1, B_2, \cdots, B_k$$

A is called a *head* and RHS of the clause is called a *body*. Some simplified examples are shown below.

```
?- grow(paris,T).
grow(X,T4) ::- temperate(X,T1).
grow(X,T5) ::- torrid(X,T2), wet(X,T3).
```

These two clauses show that the student thinks "if place X is in the temperate zone or in the torrid, wet zone then the plant grows in X". Intuitively, the clauses with the same head have a disjunctive relation and the predicates in the body have a conjunctive relation. Given a goal `grow(paris,T)` , SMDL interpreter calls the subgoals `temperate(paris,T1)` , `torrid(paris,T2)` and `wet(paris,T3)` in this order. The truth value T of `grow(paris,T)` is obtained according to $T = T_4 \vee T_5 = T_1 \vee (T_2 \wedge T_3) = true \vee (false \wedge unknown)$. Semantics of the logical operators "$\wedge$" and "$\vee$" are shown in Table 1.

2.2 Definition of SMDL

The predicate of SMDL is of the form $p(X_1, X_2, \cdots, X_m, T)$, where p is a predicate name and X_i $(1 \leq X \leq m)$ is a variable. From now on, a sequence of variables, for example $X_1, X_2, \cdots, X_m$, is abbreviated as $\tilde{X}$. T is a truth valuable or one of four truth values.

The clause of SMDL is of the form: $P::-Q_1, Q_2, \cdots, Q_m$, $m \geq 0$, where $P, Q_i (1 \leq i \leq m)$ are predicates. In the case of $m = 0$, it is called *a fact*. The SMDL program P is a finite set of clauses.

The execution process is defined as two different forms: Weak-derivation and strong-derivation. The former is like to the execution process of Prolog, that is, "a goal succeeds in weak-derivation if there exists at least one clause which derives the goal". On the other hand, the strong-derivation is somewhat different and complicated. Roughly speaking, a goal with truth value T succeeds in strong-derivation, if all the OR clauses unifiable with the goal succeed and the result of OR evaluation is T. Formal definition of the execution of G with P is given below.

[Definition 1] When a goal $G = p(\tilde{X}', T')$ is given and there exist the clause $C = p(\tilde{X}, T)::-q_1(\tilde{X}_1, T_1), \cdots, q_k(\tilde{X}_k, T_k) \in P$ and the substitution θ_0 such that $p(\tilde{X}', T') = p(\tilde{X}, T)\theta_0$, a new goal $G' = \{q_1(\tilde{X}_1, T_1), \cdots, q_k(\tilde{X}_k, T_k)\}\theta$ is derived from the goal G and the clause C. When $k \geq 1$, $\theta = \{T/(T_1 \wedge \cdots T_k)\} \cup \theta_0$. This operation is called *weak derivation* and denoted by a triple $< G', \theta, C >$. $G\theta_0\theta_1 \cdots \theta_{l-1}$ is *weakly-derived* from P, if there exists a sequence of the weak derivation: $< G, \theta_0, C_0 >< G_1, \theta_1, C_1 > \cdots < \phi, \theta_l, C_l >$, where $C_i \in P (0 \leq i \leq l-1)$. □

[Definition 2] For a given goal $G = p(\tilde{X}', T')$, if there exist a set of clauses S such that $S = \{C_i | C_i = p(\tilde{X}_i, T)::-q_{i1}(\tilde{X}_{i1}, T_{i1}), \cdots, q_{in_i}(\tilde{X}_{in_i}, T_{in_i}) \in P$ $p(\tilde{X}', T') = p(\tilde{X}_i, T_i)\theta_i$, $1 \leq i \leq m$ and the most general unifier θ_0 of $\{p(\tilde{X}')\} \cup \{p(\tilde{X}_i) | 1 \leq i \leq m\}$ then a new goal G' as shown below is derived from G and S.

$G' = \{q_{ik}(\tilde{X}_{ik}, T_{ik}) \| 1 \leq i \leq m, 1 \leq k \leq n_i\}\theta$, *where* $\theta = \{(T_i / \wedge_{k=1}^{n_i} T_{ik}) | 1 \leq i \leq m\} \cup \{T' / (\vee_{i=1}^{m} T_i)\} \cup \theta_0$.

This operation is called strong derivation and denoted by a triple $\ll G, \theta, S \gg$. $G\theta_0\theta_1 \cdots \theta_{l-1}$ is said to be *strongly-derived* from P, if there exists a sequence of the strong derivation operations as shown below,: $\ll G, \theta_0, S_0 \gg \ll G_1, \theta_1, S_1 \gg \cdots \ll \phi, \theta_l, S_l \gg$, where $S_i \subseteq P$. □

[Definition 3] When a goal $G = p(\tilde{X}', T')$ and a SMDL program P are given, the result of derivation is defined as follows. (1) $p(\tilde{X}', true)\theta$ is derived from P iff it is weakly-derived from P . (2) $p(\tilde{X}', false)\theta$ is derived from P iff it is strongly-derived from P. (3) $p(\tilde{X}', unknown)\theta$ is derived from P iff it is strongly-derived from P. (4) Otherwise, $p(\tilde{X}', fail)\theta$ is derived from P. □

3 HSMIS

In general, an inductive inference algorithm is based on the assumption that all the observed data(oracles) are consistent. In our case, it requires that student's answers are consistent. Unfortunately, however, the assumption does not always hold. Students change their minds and sometimes make careless mistakes. Therefore, a student model inference system must cope with inconsistent data. In HSMIS, ATMS is employed for this purpose. Fig. 1 shows the block diagram of HSMIS. It consists of SMIS[Ikeda *et al.*, 1989], ATMS, Virtual oracle generator(explained below) and Contradiction resolving system(CRS). The main task of ATMS is to manage consistency of a set of assumptions (environment) used by the problem solver, SMIS in our case. Virtual oracle generator is responsible for improving the performance of model inference by generating assumed student answers based on the reliability of the student without asking

Table 1: Definition of $\wedge$ and $\vee$.

(a) $\wedge$ operator

$\wedge$	true	unk.	false	fail
true	true	unk.	false	fail
unk.	unk.	unk.	false	fail
false	false	false	false	fail
fail	fail	fail	fail	fail

(b) $\vee$ operator

$\vee$	true	unk.	false	fail
true	true	true	true	true
unk.	true	unk.	unk.	fail
false	true	unk.	false	fail
fail	true	fail	fail	fail

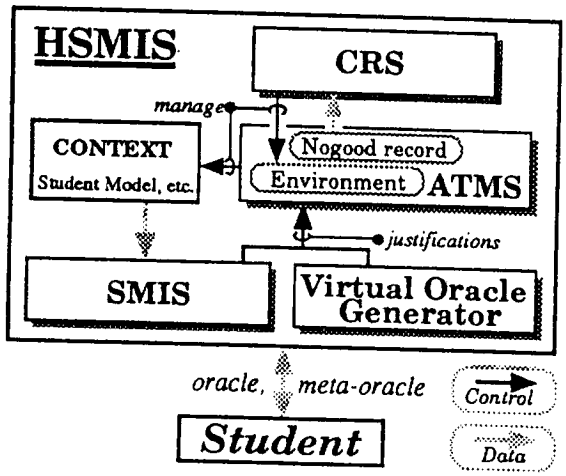

Figure 1: Block diagram of HSMIS.

the student questions. CRS resolves the inconsistency identified by changing the environment.

3.1 SMIS

A pair of a problem and an answer to it is called an oracle and is used as data to be covered by the model obtained.

[Definition 4] An oracle is of the form $< p(\tilde{X}', T), T' >$, where $\tilde{X}'$ is a sequence of ground terms, T is a truth variable and $T' \in \{true, false, unknown\}$. □

Because *fail* represents the system's understanding of student, it cannot be the truth value of an oracle. A set of oracles given to the system is called an *oracle set* and denoted by Ω.

[Definition 5] The clause $C = p(\tilde{X}, T)$::-$q_1(\tilde{X}_1, T_1)$, $q_2(\tilde{X}_2, T_2), \cdots, q_k(\tilde{X}_k, T_k) \in P$ covers $p(\tilde{X}', T')$, when there exist θ such that $C\theta = p(\tilde{X}', T')$::-$q_1(\tilde{X}'_1, T'_1)$, $q_2(\tilde{X}'_2, T'_2), \cdots, q_k(\tilde{X}'_k, T'_k)$, $T' = \bigwedge_{i=1}^{k} T'_i$ and $\{< p(\tilde{X}', T), T' >\} \cup \{< q_i(\tilde{X}'_i, T_i), T'_i > | 1 \leq i \leq k\} \subset \Omega$. We call the $q_1(\tilde{X}'_1, T'_1), \cdots, q_k(\tilde{X}'_k, T'_k)$ a *top-level trace* of C for $p(\tilde{X}', T')$. □

A simplified but concrete example of top-level trace is shown in Fig.2, where the oracles $O_1, \cdots, O_6$ correspond to student's answers (a) through (f), respectively. In this case, the clause C covers O_1, where the top-level trace is made by O_2 and O_3.

SMIS applies the following procedure repeatedly to the model: (1) if there is a difference between an oracle and the fact derived from the student model, activate the student model diagnosis system, SMDS, to identify the cause of the difference. (2) According to the diagnosis, SMIS selects an appropriate operation, either removal of an incorrect clause or addition of a new clause, and informs ATMS of it.

3.1.1 SMDS: Student Model Diagnosis System

SMDS traces derivation process of the current model and checks the results with oracles. SMDS has three subprocedures, such as, *ip* , *fp* and *failp* . The procedure *ip* finds out where a new clause should be added. The procedure *fp* detects an incorrect clause which should be removed from the model. The procedure *failp* dynamically decides which procedure, *fp* or *ip* , should be activated. SMDS selectively activates one of them according to the difference between the oracle's truth value and the one derived from the student model.

[Definition 6] The model P is said to be *weak*, if there exists an oracle $< p(\tilde{X}', T), T' >$ and $p(\tilde{X}', T')$ is not weakly-derived from the current model P. □

The cause of the *weakness* is identified by the procedure *ip* . To cover the uncovered goal detected by *ip* , HSMIS searches for a new clause to add into the model.

We define $\prec$ to be the binary relation over truth values. When two clauses derive different truth values, T_1 and T_2, for a goal, T_1 is dominant if the relation $T_1 \succ T_2$ holds.

[Definition 7] $T_1 \succ T_2$ iff $T_1 = T_1 \vee T_2$, where $T_1, T_2 \in \{true, false, unknown\}$ and $T_1 \neq T_2$. □

[Definition 8] Assume that $< p(\tilde{X}', T), T' > \in \Omega$ and $T' \in \{unknown, false\}$. The model P is said to be *strong* if $p(\tilde{X}', Tg)$ such that $Tg \succ T'$ is weakly-derived from P. □

A strong model has at least one incorrect clause which has a refutation as defined below.

[Definition 9] The clause $C = p(\tilde{X}, T)$::-$q_1(\tilde{X}_1, T_1)$, $\cdots, q_k(\tilde{X}_k, T_k)$ is said to be *incorrect*, if there exist $< p(\tilde{X}', T), T' > \in \Omega$ and $\{< q_i(\tilde{X}'_i, T_i), T'_i > | 1 \leq i \leq k\} \subset \Omega$, where $T' \prec \bigwedge_{i=1}^{k} T'_i$. $p(\tilde{X}, T')$::-$q_1(\tilde{X}'_1, T'_1)$, $\cdots, q_k(\tilde{X}'_k, T'_k)$ is called *refutation* for C. □

In Fig.2, the clause C is refuted by oracles O_4 , O_5 and O_6. The incorrect clause with a refutation, which should be removed from the model as a cause of *strongness*, is identified by the procedure *fp* .

[Definition 10] The model P is said to be *incomplete* if $< p(\tilde{X}', T), T' > \in \Omega$ and $p(\tilde{X}', fail)$ is derived from P □

The procedure *failp* identifies the cause of the *incompleteness*, which is either an uncovered goal or an incorrect clause. Because of the space limitation, detailed explanation of the procedures, *ip* , *fp* and *failp* , is omitted.

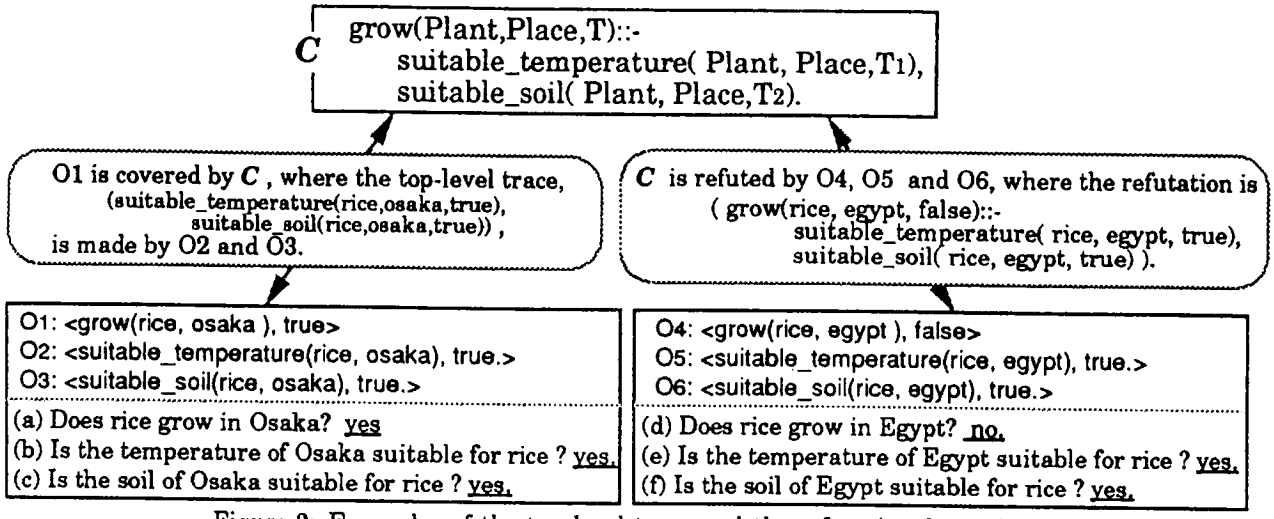

Figure 2: Examples of the top-level trace and the refutation for a clause.

More detailed explanation and Prolog implementation of each procedure is shown in [Ikeda *et al.*, 1992].

3.1.2 Search for a new clause

A clause to be added into the model is searched by using the refinement graph having the clause as a node according to breadth-first search. The directed arc $C \xrightarrow{\rho} C'$ on the refinement graph suffices the following relation.

$C \xrightarrow{\rho} C'$: When the clause C' covers a goal A
then the clause C covers A

With contraposition of this condition, the search on the refinement graph can be pruned. That is, when the clause does not cover A, branches leading to descendents of C in the graph can be pruned. A refinement operator ρ , which is defined by modifying that defined in MIS, produces a refined clause C' from the given clause C.

A refinement graph represents knowledge which enables efficient search for a clause to be added into the model. However, it does not have any a priori knowledge of bugs. So, it always tries to find out a clause from a fixed root, that is, the most general clause, independently of materials. Note here that we can introduce the concept of bug when we know the material well. Given some typical bugs specific to the teaching material under consideration, search procedure can begin searching from these bugs, which makes search very efficient.

3.2 Managing consistency of the inference process

There are two types of nonmonotonicity in our system. One is inherent to inductive model inference itself and the other is caused by inconsistency of students' behavior. Both of them are formalized in a unified architecture on HSMIS by combining SMIS and ATMS. The modeling task performed by HSMIS is defined as follows:

[Definition 11] When an oracle set Ω is given, infer a model for $\tilde{\Omega}$ allowing the distance $D(\Omega, \tilde{\Omega})$ to be minimized. $\tilde{\Omega}$ satisfies the following restriction.

$\tilde{\Omega} = \{< A_i, \tilde{T}_i > \mid 1 \leq i \leq k\}$
for $\Omega = \{< A_i, T_i > \mid 1 \leq i \leq k\}$. □

The distance $D(\Omega, \tilde{\Omega})$ is defined as number of the oracles of Ω whose truth value is different from the corresponding one in $\tilde{\Omega}$.

With the aid of schematic diagram shown in Fig. 1, the overall behavior of the system will be made clear in this subsection. (1) Given student answers ("real oracle"), Virtual oracle generator generates virtual oracles if necessary and pass them to ATMS with the real oracles. (2) SMIS informs ATMS of all the inference process. When a contradiction is informed, ATMS computes the label responsible for the inconsistency based on the information given up to that point and store it in the nogood record. (3) SMIS asks CRS to resolve the inconsistency. (4) According to the cause of inconsistency identified, CRS selects a new environment which is consistent by asking ATMS to check its consistency. (5) ATMS answers the queries by inspecting the nogood record and (6) pass the control to SMIS together with a new context supported by the new consistent environment.

With regard to the contradiction caused by the inconsistent oracle set, CRS searches for a consistent $\tilde{\Omega}$ in the ascending order of $D(\Omega, \tilde{\Omega})$. To improve the efficiency and educational validity of selecting the consistent oracle set which has the minimal distance, we incorporated some domain-independent heuristics into the search procedure: Give priority to correct answers, give priority to the oracles supporting the plausible clause, which is supported by relatively many oracles, and so on. After that, HSMIS continues inference on the new consistent environment including $\tilde{\Omega}$. Details of HSMIS reasoning process are explained in the following subsections.

3.2.1 Description of inference process

Conditions for HSMIS to add a new clause $C = p(\bar{X}, T) ::\text{-} q_1(\bar{X}_1, T_1), \cdots, q_k(\bar{X}_k, T_k)$ to a model can be described as shown below.

if cond1: $(< p(\bar{X}', T), T' > \in \tilde{\Omega})$ **and**
cond2: (C is correct) **and**
cond3: (C covers $p(\bar{X}', T')$) **and**
cond4: (C is the most general clause)
then (add C to the model)

The conditions, i.e. cond1 through cond4, are described as follows.

cond1: To cope with nonmonotonicity of student's answers, the oracle is dealt with as the following assumed node of ATMS.

$$< oracle(p(\tilde{X}', T), T'), \{\{a_i\}\}, \{\{a_i\}\} >^1$$

cond2: The correctness, which means there is no refutation for C, is also dealt with as the assumption node as shown below.

$$< correct(C), \{\{a_i\}\}, \{\{a_i\}\} >$$

cond3: The condition means that C has a correct top-level trace for $p(\tilde{X}', T')$ in $\tilde{\Omega}$. HSMIS informs ATMS of its existence in the following form.

$$\left.\begin{array}{l} oracle(p(\tilde{X}', T), T') \\ oracle(q_1(\tilde{X}'_1, T_1)\theta, T'_1) \\ \quad\vdots \\ oracle(q_k(\tilde{X}'_k, T_k)\theta, T'_k) \end{array}\right\} \Rightarrow cover(C, p(\tilde{X}, T'))$$

where $T' = \bigwedge_{i=1}^{k} T'_i$ and $(q_1(\tilde{X}_1, T_1), \cdots, q_k(\tilde{X}_k, T_k))\theta$ is a correct top-level trace of the clause C for $p(\tilde{X}', T')$.

cond4: When C is added to the model, it has to be guaranteed that its ancestor clause in the refinement graph, which is more general than C, does not exist in the current model. However, an ancestor clause may be added to the model as the inference proceeds, because of the nonmonotonicity of the inference process. Therefore, the generality of C is also dealt with as the assumption.

$$< general(C), \{\{a_i\}\}, \{\{a_i\}\} >$$

Addition of C to the model is informed ATMS in a style as shown below.

$$\left.\begin{array}{l} oracle(p(\tilde{X}', T), T') \\ correct(C) \\ cover(C, p(\tilde{X}', T')) \\ general(C) \end{array}\right\} \Rightarrow model(C)$$

If the environment is changed and the $model(C)$ does not hold in the new environment, ATMS changes the status of the node from *in* to *out* .

When a refutation for a clause C identified by the SMDS, it is informed ATMS in the following form.

$$\left.\begin{array}{l} oracle(p(\tilde{X}', T), T') \\ oracle(q_1(\tilde{X}'_1, T_1), T'_1) \\ \quad\vdots \\ oracle(q_k(\tilde{X}'_k, T_k), T'_k) \end{array}\right\} \Rightarrow refutation(C))$$

where $p(\tilde{X}', T')$::-$q_1(\tilde{X}'_1, T'_1), \cdots, q_k(\tilde{X}'_k, T'_k)$ is a refutation for C.

When it is found that C' does not cover $p(\tilde{X}', T')$, HSMIS informs ATMS of it as an assumption.

$$< uncover(C', p(\tilde{X}', T')), \{\{a_i\}\}, \{\{a_i\}\} >$$

3.2.2 Controlling the modeling process

HSMIS tries to model the student from her behavior during which it automatically asks questions which contribute to disambiguation of alternative clause selection and diagnosis. In other words, HSMIS asks questions regardless of their appropriateness in the sense of tutoring. This requires some control mechanism of the HSMIS behavior. This subsection describes several additional mechanisms introduced to augment the HSMIS.

Virtual Oracles: Let us discuss the initial model problem. There are two alternative initial models: one is empty which means the teacher does not know anything about the student in advance and the other is complete knowledge (teaching material) which means teacher assumes the students usually understand the material very well. Although the former case is reasonable, the system tends to ask many questions to get a lot of information of how well the student understands the material. On the other hand, the latter case does not require many questions at least for excellent students, since the model can explain their correct behavior. This characteristics is very reasonable in real tutoring. Therefore, we decided to employ the latter. However, a serious problem still remains. One cannot simply put a clause into the student model without any justification.

In order to cope with this problem, we devised an Virtual oracle generator, which generates plausible student answers based on the reliability of the current student model instead of asking questions. When a student's behavior is confined within the scope of her teacher's prediction, the teacher asks less questions by replacing the necessary information with correct answers. We call this type of oracle a "virtual oracles".

Let C be an SMDL clause $p(\tilde{X}, T)$::-$q_1(\tilde{X}_1, T_1), \cdots, q_k(\tilde{X}_k, T_k)$ which is either correct knoledge or plausible buggy knoledge in the teaching material. When the student makes an answer $p(\tilde{X}', T')$ for the head of C and C is supported by $p(\tilde{X}', T')$ and a set of correct answers concerning C, ATMS is informed of an assumtion $trust(C)$, which means "C is reliable." When trust(C) is *in* the current environment, Virtual oracle generator generates a set of virtual oracles, that is, $\{< q_1(\tilde{X}'_1, T_1), T'_1 >, \cdots, < q_k(\tilde{X}'_k, T_k), T'_k >\}$. The virtual oracles together with $p(\tilde{X}', T')$ construct a correct top-level trace of C and are correct answers of the teaching material. For each virtual oracle $< q_i(\tilde{X}'_i, T_i), T'_i >$, ATMS is informed its generation according to the following justification:

$$trust(C), oracle(p(\tilde{X}', T), T') \Rightarrow v\_oracle(q_i(\tilde{X}'_i, T_i), T'_i))$$

SMIS treats *oracles* and *v_oracles* in the same manner, while ATMS manages their consistency.

Meta-Oracles: Students sometimes want to say her knowledge in the form of knowledge instead of facts. And the system sometimes wants to ask the student the reason why she answers a question that way. The following is an example.

```
System : Does rice grow in Russia?
Student: Yes, it does.
System : Why do you think rice grows in Russia?
Student: It has wide flat field and river.
```

In this case, HSMIS can obtain an oracle and a clause as follows.

$$< grow(rice, russia, T), true >$$

[1] An ATMS node is a triple $< D, L, J >$, where D is the datum used in the reasoning system, L is a label, which is a set of environment the dautm holds and J is a set of justifications. An assumed node has the same set composed of a single identifier as a label and a justification(in this paper, the identifier of the assumption will be expressed with a_i).

```
grow(rice, Place) ::-
        flat_field(Place),
        river(Place).
```

The clause obtained from the student are called "meta-oracle". When the clause C is added to the model based on a "meta-oracle", HSMIS informs ATMS of the following justification.

$$\left.\begin{array}{l} metaOracle(C, yes) \\ correct(C) \\ general(C) \end{array}\right\} \Rightarrow model(C)$$

Similarly, when the clause C is removed from the model based on a "meta-oracle", HSMIS informs ATMS of the following justification.

$$metaOracle(C, no) \Rightarrow refutation(C)$$

The meta-oracle is dealt with as the following assumption node of ATMS.

$$< metaOracle(C, V), \{\{a_i\}\}, \{\{a_i\}\} >$$

3.2.3 Detection of contradiction

The contradiction derived during the inference process of HSMIS is classified into the following seven types: 1) Contradiction of correctness, 2) Contradiction of cover test, 3)Contradiction of clause generality, 4) Contradiction of trust, 5) Contradiction of meta-oracle, 6) Contradiction of oracle and 7) Failure of clause search. More detailed explanation of the contradictions and the mechanism to detect them is shown in [Ikeda *et al.*, 1992]. When any types of the contradiction is detected, ATMS is informed of it and updates the nogood record.

3.2.4 Resolution of contradiction

$correct(C)$, $uncover(C, A)$, $general(C)$, $trust(C)$ and $metaOracle(C)$ are called to be *default assumptions*, which means that "so long as no contrary evidence is found, it is assumed to be *in* ". The set of these assumptions included in the environment is called a *default environment* (expressed with D_e). From this, the current environment C_e can be expressed with

$$C_e = De \cup \tilde{\Omega}$$

Thus, contradiction can be classified into I) one regarding the default environment and II) one regarding the oracle environment. I) is the contradiction caused by the nonmonotonicity inherent to the inference process, whereas II) is the one caused by the nonmonotonicity of the student's answers.

I) A resolution method for contradiction of the default environment can be easily derived from its definition.

Suppose that contradiction is detected since there exist both the $refutation(C)$ and the $correct(C)$ in the current context. As has already been stated, the $correct(C)$ can be hold so long as there does not exist the refutation for the clause C ($refutation(C)$). Therefore, the inconsistency can be resolved by removing the $correct(C)$ from the default environment. Similarly, the assumptions, $uncover(C, A)$, $trust(C)$ and $metaOracle(C)$, are removed from the default environment when the contradictory data is found.

II) In the case of the oracle contradiction, it is necessary for the distance between the two oracle sets to be minimized and all the contradiction already detected will be evaded in the new environment. This is done by the heuristic search method.

3.3 Behavior of HSMIS

In this subsection, we show an example of how HSMIS works, in which the description is partially simplified because of space limitation, for example, the description of teaching material is not correct in the strict sense(for example, the first argument of grow is omitted).

[An example of HSMIS modeling process]

Does rice grow in Japan? → grow(Place)::- temp(Place), soil(Place). *C1*

yes. o1: grow(japan), yes.
v_o2: tmp(japan), yes.
v_o3: soil(japan), yes.

*Because o1 is a correct answer, the correct clause **C1 is added** to the model. And then, v_o2 and v_o3, which are correct answers, are generated by oracle-generator. They are justified by trust(C1) and oracle(o1). ATMS is informed of the following assumptions and justifications.*

Assumptions: oracle(o1), trust(C1), correct(C1), general(C1)

Justifications:

trust(C1), oracle(o1) ⇒ v_oracle(v_o2)
trust(C1), oracle(o1) ⇒ v_oracle(v_o3)
oracle(o1), v_oracle(v_o2), v_oracle(v_o3) ⇒cover(C1,o1).
oracle(o1), correct(C1), cover(C1,o1), general(C1)⇒model(C1)

Does rice grow in Kiev? → grow(Place)::- soil(Place). *C2*

yes. o4: grow(kiev), yes.
v_o5: soil(kiev), yes.

*A buggy clause **C2 is added** to the model, because o4 is a typical incorrect answer(Kiev is too cold for rice to grow). A virtual oracle v_o5, which is justified by trust(C2) and oracle(o4), is generated by the oracle-generator. Since the clause C1 is a descendent node of C2 in refinement graph, **model(C1) becomes 'out'** by outing general(C1). Then the status of v_o2 and v_o3 becomes 'out'. However, v_o3 becomes 'in' at once, because it is justified by trust(C2) and oracle(o1).*

Assumeptions oracle(o4) , trust(C2), correct(C2),general(C2)

Justifications:

trust(C2), oracle(o1) ⇒ v_oracle(v_o3)
trust(C2), oracle(o4) ⇒ v_oracle(v_o5)
oracle(o4), v_oracle(v_o5) ⇒cover(C2,o4).
oracle(o4), correct(C2), cover(C2,o4), general(C2)⇒model(C2)

Oracle environment

in	o1:<grow(japan),yes. >	*in*	o4:<grow(kiev), yes. >
out	v_o2:<tmp(japan), yes. >	*in*	v_o5:<soil(kiev), yes. >
in	v_o3:<soil(japan), yes. >		

Does rice grow in Moscow?

no. o6:grow(moscow), no.

Is Soil of Moscow suitable for rice to grow?

yes. o7:soil(moscow), yes.

SMDS: C2 is refuted by o6 and o7 ; Refutation: *grow(moscow,false)::-soil(moscow,true)*

***C2 is removed** because SMDS found out its refutation. A contradiction is found between refutation(C2) and correct(C2). CRS resolves this by outing the default assumption correct(C2). Then the model(C2), oracle(v_o3) and oracle(v_o5) become 'out'. HSMIS starts to search for a clause which covers o4.*

Justifications:

oracle(o6), oracle(o7)⇒refutation(C2).

Oracle environment

in	o1:<grow(japan),yes. >	*in*	v_o5:<soil(kiev), yes. >
out	v_o2:<tmp(japan), yes. >	*in*	o6:<grow(moscow), no. >
out	v_o3:<soil(japan), yes. >	*n*	o7:<soil(moscow), yes. >
in	o4:<grow(kiev), yes. >		

Is soil of Kiev suitable for rice to grow?

yes. o5:soil(kiev), yes. (confirmed by the student)

Is Temperature of Kiev suitable for rice to grow?

no. o8: temp(kiev), no.

SMIS: C3: grow(Place):-temp(Place) cannot cover grow(kiev,yes).
Contradiction is found, because HSMIS has failed to search for a cover of grow(kiev,true). CRS changes the oracle environment to resolve the contradiction based on give-a-priority-to-correct-answer heuristics. In this case, o4 becomes 'out' and o9, whose truth value is opposite to o4, is believed according to give-priority-to-correct-answer heuristic. Finally, ***C1 is added*** *to the current model (model(C1) becomes 'in' again) in the new consistent oracle environment .*
< Assumptions and Justifications are omitted >
Oracle environment

in	o1:<grow(japan),yes. >	*in*	o6:<grow(moscow), no. >
in	v_o2:<tmp(japan), yes. >	*in*	o7:<soil(moscow), yes. >
in	v_o3:<soil(japan), yes. >	*in*	o8:<temp(kiev), no. >
out	o4:<grow(kiev), yes. >	*in*	o9:<grow(kiev), no. >
in	o5:<soil(kiev), yes. >		

In order to realize the above behavior of HSMIS, at least two kinds of knowledge is required. One is the knowledge for refinement graph generation. In the above example, for example, the following knowledge is used for generating three clauses $C1$, $C2$ and $C3$.

$$declare\_called(grow(Place,T), \\ [temp(Place,T_1), soil(Place,T_2)])$$

This means that the predicates in the second argument, that is, $temp(Place,T_1)$ and $soil(Place,T_2)$, can appear in the body of a clause whose head is $grow(Place,T)$. If this form of knowledge including necessary predicates is prepared, HSMIS can automatically generate a complete set of clauses as a model of the student.

The other one is the correct domain knowledge which provides HSMIS with correct answers. In the above example, the knowledge is used for three purposes, that is, for the virtual oracle generation, the environment management by CRS, and the problem generation.

In the above example, the student model changed two times. The first change from $C1$ to $C2$ can be regarded as corresponding to the nonmonotonicity inherent to the inference process, that is, $C1$ is not an appropriate hypothesis of the student understanding at that time. Meanwhile, the second change from $C2$ to $C1$ corresponds to the nonmonotonicity of student's understanding.

4 Concluding remarks

Both SMDL interpreter and HSMIS have been implemented in Common ESP(Extended Self-contained Prolog) and incorporated in a framework for ITS, called FITS[Mizoguchi and Ikeda, 1991]. HSMIS can cope with a variety of teaching materials as far as it can be represented in Prolog. Therefore, the authors think that the generality of HSMIS is relatively high.

Two ITS's have been built using FITS, one is on geography and the other is on chemical reactions. Simple examination of both systems shows they run in real time if the oracle contradiction does not occur. When the oracle contradiction occurs and there is no heuristics for its resolution, we find that the cost to resolve the contradiction is very expensive.

The current implementation of contradiction resolution is based on a somewhat brute-force method using domain-independent heuristics. To improve the efficiency, more powerful domain-dependent heuristics should be employed. It seems promising to adopt a belief revision mechanism developed by X. Huang et.al, which provides efficient, minimal revision of belief bases by using attention(focus) in the belief space[Huang *et al.*, 1991b].

We have discussed mechanisms to avoid inconsistency in model inference in this paper. However, there exist such students who have contradictions in their head. To cope with modeling of such students, the system may not avoid the inconsistency but has to model inconsistent knowledge as it is. This issue will involve the development of more sophisticated control mechanism for student modeling. We are currently engaging in the issue, where student's inconsistent knowledge is modeled in multiple worlds which are again supported by ATMS[Kono *et al.*, 1992].

Acknowledgements : The authors are grateful to reviewers for their valuable comments.

References

[Burton, 1982] R. R. Burton. In D. H. Sleeman, editor, *Intelligent Tutoring Systems*, pages 157–183, Academic Press, 1982.

[de Kleer, 1986] J. de Kleer. An Assumption-based Truth Maintenance System. *Artificial Intelligence*, 28:127–162, 1986.

[Huang *et al.*, 1991a] X. Huang, G.I. McCalla, J. E. Greer, and E. Neufeld. Revising Deductive Knowledge and Stereotypical Knowledge in Student Modeling. *User Modeling and User-Adapted Interaction*, 1:87–115, 1991.

[Huang *et al.*, 1991b] X. Huang, G.I. McCalla and E. Neufeld. Using Attention in Belief Revision. *In Proceedings of the Ninth National Conference on Artificial Intellegeince*, pages 275–280, 1991.

[Ikeda *et al.*, 1988] M. Ikeda, R. Mizoguchi and O. Kakusho. A Hypothetical Model Inference System. *Trans. of IEICE Japan*, J71-D:1761–1771, 1988 (in Japanese).

[Ikeda *et al.*, 1989] M. Ikeda, R. Mizoguchi and O. Kakusho. Student Model Description Language SMDL and Student Model Inference System. *Trans. of IEICE Japan*, J72-D-II:112–120, 1989 (in Japanese).

[Ikeda *et al.*, 1992] M. Ikeda, Y. Kono and R. Mizoguchi. Nonmonotonic Model Inference. *Technical Report of Artificial Intelligence Research Group, ISIR Osaka Univ.*, AI-TR-92-10, 1992.

[Kono *et al.*, 1992] Y. Kono, M. Ikeda and R. Mizoguchi. To contradict is human — Student modeling of inconsistency —. In C. Frasson, G. Gauthier, and G. I. McCalla, editors, *Intelligent Tutoring Systems*, ITS '92, Montréal, Canada, Proceedings, pages 451–458. LNCS 608, Springer-Verlag, 1992.

[Mizoguchi and Ikeda, 1991] R. Mizoguchi and M. Ikeda. A Generic Framework for ITS And Its Evaluation. In R. Lewis and S. Otsuki, editors, *Advanced Research on Computers in Education*, pages 63–72. North-Holland, 1991.

[Shapiro, 1982] E. Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1982.

[Wenger, 1987] E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, California, 1987.