Contents

List of Figures

List of Tables

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 a snapshot of the 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 ITSs. However, all the methods have some limitations and no complete and sound inference procedure for modeling has been obtained yet. In this paper, we formulate the student modeling problem as an inductive inference problem, i.e., a problem of constructing a model explaining observed data which are, in our case, student's answers to the problems given.

A student who is in the fixing stage[1] of the acquired knowledge often shows contradictory behavior. This means a student is apt to apply problem solving methods unstably, since he[2] has not built them or since he has not completed in a formulation of related concepts, etc. It is clear that he shows such nonmonotonic learning processes in fixing his knowledge in the process of acquiring new knowledge. There are other types of contradiction to be considered in designing a student modeling system. A modeling system often gets an answer from the student that is consistent with his current belief but inconsistent with his past answers, because he has changed his mind as his learning proceeds or he sometimes makes careless mistakes, so-called "slips."

Contradictions which a modeling system should cope with are classified into the following two types:

(1) contradictions which should be resolved by revising the student model, and
(2) contradictions which should be captured as they are.

Generally speaking, an ITS should follow a student's nonmonotonic change. Furthermore, a student modeling system should realize more flexible modeling behavior and construct reasonable student models from didactic viewpoints by embodying a teacher's insight, e.g., the ability to capture her student's status by asking fewer questions. Contradictions of type (1) inevitably appear in the student modeling process. Therefore, a student modeling system is required to have the ability to cope with various kinds of nonmonotonicities. To this end, a student modeling system should always make belief revisions to keep data for inference consistent. Surprisingly, however, only Huang *et al* (1991a;1991b) has tackled this problem, except that Woolf *et al* (1993) has pointed out its significance. The authors have been attacking this issue and developing an inductive student model inference algorithm HSMIS (Kawai *et al*, 1987; Mizoguchi *et al*, 1987; Mizoguchi *et al*, 1991). HSMIS (Hypothetical Student Model Inference System) employs the ATMS (Assumption-based Truth Maintenance System) (deKleer, 1986) to maintain consistency of the student modeling process (Ikeda *et al*, 1988). The architecture of HSMIS is based on a logic-based inductive inference algorithm SMIS (Ikeda *et al*, 1989), whose model description language is also a logic-based language called SMDL (Student Model Description Language) which takes four truth values to represent a student's understanding. This paper first discusses the characteristics of contradictions and nonmonotonicities observed in the student modeling process. Then a student modeling architecture to deal with those nonmonotonicities is presented, after SMDL and SMIS are defined. Thus HSMIS realizes relatively high model representation power and modeling ability.

The second problem, that is, to capture a student's contradictory knowledge as it is, seems more important from educational viewpoints. The Socratic method, for example, is a contradiction-based tutoring strategy which teachers use especially to help students in the fixing stage. It is a well-known and already verified method that gives such a student a strong impression that he misapplied his knowledge. Although building high-fidelity student models is an intractable problem (Self, 1988), an ITS should have a student model which is precise enough to handle tutoring strategies integrated into the ITS (Dillenbourg, 1989). In order to generate sophisticated tutoring behavior like the Socratic method, student modeling methods should be able to cover a student's contradictory knowledge. His knowledge acquisition and fixing processes should be captured by modeling him as he is, even if he has contradictory knowledge (Kono *et al*, 1992; Kono *et al*, 1993b; Kono *et al*, 1994).

[1] "Fixing stage" means an intermediate learning stage where acquired knowledge is not completely established yet.
[2] For simplicity "she" is used to refer to the teacher and "he" to refer to the student in general.

Some might say that some ITSs, e.g., WHY (Stevens *et al*, 1977), have realized the Socratic method. Tutoring behaviors generated by most Socratic tutors, however, do not notice or suggest contradictions that exist inside a student in spite of the fact that such a suggestion is the real behavior of the Socratic method. They only give a new negative example of his knowledge. They do not realize the real and complete Socratic method. HSMIS is equipped with a mechanism to cope with all the contradictions of type (1) which are formulated in this paper. HSMIS generates student models in SMDL which has more expressiveness than Horn clauses. Therefore, HSMIS realizes a high modeling ability and model expressiveness which surpasses most other modeling methods.

This paper presents a new methodology for handling a student's contradictory knowledge in conjunction with Socratic tutoring based on the concrete foundation of student modeling. It is difficult for not only modeling systems but also human teachers to distinguish the two types of contradiction, i.e., one where a modeling system should treat the student's knowledge as contradictory and the other where the system should follow the student's nonmonotonic change, because all of their indications are very similar. Two types of contradictions are formulated. Type (1) is named "single world contradiction" which is coped with by HSMIS and type (2) is named "multi-world contradiction" which is coped with by the structure of a concept discrimination tree. The two methodologies, HSMIS and a multi-world inference mechanism, are successfully incorporated into the architecture of THEMIS by enumerating and equipping domain independent heuristics to distinguish the two types of contradiction. In the newly formulated THEMIS, ATMS plays another important role of managing multiple worlds which enable the modeling of students with contradictions.

This paper first classifies and formulates the above two types of contradiction which should be dealt with in student modeling. HSMIS, a nonmonotonic inductive student modeling system, which is able to cope with single world contradictions is then formulated. Next, a mechanism which copes with multi-world contradictions is formulated, which is incorporated into THEMIS based on HSMIS. Finally, THEMIS is evaluated and compared with other representative systems.

2 Contradictions in student modeling

A student's answer to a question is represented by a pair of a fact and its truth value, and is called an oracle. The student modeling problem can be formulated as an inductive inference problem, i.e., a problem of constructing a model explaining observed data. A set of oracles acquired by observation of a student's behavior within a certain period tends to be inconsistent for several reasons. Such inconsistent behavior is classified into the following three types of contradiction according to the causes of them:

[A1] *Oracle contradictions caused by change of student's mind*: A student's learning process is essentially attained by acquiring new knowledge causing change of his mind. The consistency of his answers within the whole learning process can be easily lost, because he behaves based on his current knowledge independent of his previous knowledge.

[A2] *Oracle contradictions caused by slips*: A student often makes careless mistakes. Oracles based on them are inconsistent with his actual knowledge, hence, the set of oracles that contains slips is inconsistent.

[A3] *Student knowledge contradictions*: A student sometimes has inconsistent knowledge in his head which also causes contradictory oracles.

These three kinds of contradiction are referred to as "student contradictions", since they are related to nonmonotonicities of a student's behavior or knowledge.

The student modeling process is essentially hypothetical, hence, the completeness of an inferred student model is not always guaranteed. The expectation of a student's answer deduced from the current student model is often different from new oracles, when the current model does not completely represent his current status. Assumptions which were assumed when the current model was inferred become inconsistent with the set of oracles. Such a type of contradiction is called

[B] *assumption contradiction in modeling.*

Contradictions of type [A1], [A2] and [B] should be placed in the same category when classifying them according to how to treat them, because all of them should be resolved by revising the current

Question 1
A sphere has passed through the origin with velocity 19.6m/s to the right at $t = 0$s. It continues linear and uniformly accelerated motion along x-axis of a horizontal plane, and stopped at $t = 2$s. It mass = 2kg. Get its displacement at $t = 1$s. Determine both the direction and the magnitude of the applied force.
Student's answer
$a = \frac{v_t - v_0}{T_t - T_0} = \frac{0-19.6}{2-0} = -9.8\text{m/s}^2$
$S_t = S_0 + v_0 t + \frac{1}{2}at^2 = 0 + 19.6*1 + \frac{-9.8*1^2}{2} = 14.7$
The displacement is 14.7m from the origin.
$F = ma = 2 * -9.8 = -19.6\text{kg}\cdot\text{m/s}^2$
It receives 19.6 $\text{kg}\cdot\text{m/s}^2$ of force to the left.

Question 2
A sphere has passed through the origin with velocity 19.6m/s to the right at $t = 0$s along x-axis of a horizontal plane. P is moving with linear motion with its velocity decreasing at a uniformed rate. It stopped at $t = 2$s. Determine its displacement at $t = 1$s.
Student's answer
$S_t = S_0 + vt = 0 + 19.6*1 = 19.6$
The displacement is 19.6m.

Question 3
An sphere is thrown directly upwards with initial velocity 19.6m/s at t=0s. It reaches at the maximum hight at t=2s. Determine the direction of the applied force at t=1s.
Student's answer
$v_t = \frac{v_{t'} - v_0}{T_{t'} - T_0} * t + v_0 = \frac{19.6-0}{2-0} * 1 + 0$
$= 9.8\text{m/s}$
It is still moving upward at $t = 1$s.
Therefore, it is still receiving upward force at $t = 1$s.

Figure 1: Examples of behavior of a student who has undifferentiated concepts.

student model. On the other hand, contradictions of type [A3] should not be resolved, but a student's contradictory knowledge should be represented as it is so as to utilize them effectively in tutoring.

Some examples of student knowledge contradictions are given in Section 2.1, and a formulation of the contradictions is made based on the multi-world logic in subsequent subsections. All types of the above contradictions in student modeling are then classified and discriminated from the viewpoint of how they should be handled in the student modeling process.

2.1 What is a tudent knowledge contradiction?

Let us assume that a student is in the stage of acquiring a certain new concept and that he has not fully discriminated it from other related concepts he has already acquired. Such a student is apt to behave unstably in applying knowledge to solve problems which contain the undifferentiated concept. Figure 1 indicates the behavior of a student who has undifferentiated concepts, i.e., the concept of "uniform motion" and the concept of "uniformly accelerated motion." He correctly calculated the location of P in Question 1, which specifies the type of motion as "linear and uniformly accelerated motion." In Question 2, however, he mistook a uniformly accelerated motion for a uniform motion, and so applied problem solving knowledge for uniform motion. Such a situation occurs due to his confusion between the two concepts. As a result, his problem solving ability becomes unstable.

A student can choose certain problem solving methods appropriate for the problem given, if he has well-discriminated concepts and has adequate knowledge of their attributes. If he has not, however, he might misapply a procedure which belongs to another world by taking no notice of particular attributes of the problem. For instance, methods to "calculate the location of a moving object" are associated with both concepts, such as uniformly accelerated linear motion ($S_t = S_0 + v_0 t + at^2$ or $S_t = S_0 + (v_0 + v_t) * t/2$) and uniform motion ($S_t = S_0 + vt$). In solving Question 2 in Figure 1, he retrieves the method defined in the concept of uniform motion, while he should apply the method defined in the concept of uniformly accelerated linear motion.

A more interesting example of student knowledge contradiction is found in Question 3 in Figure 1. The student who had correctly calculated the force that the sphere receives in Question 1 could not determine the correct direction of the force in Question 3 in spite of the fact that the two motions are physically identical except for the direction of the motion.

Such conflicts among his answers suggest the "multi-world inference" assumption that he partitions his whole storage and inference space. Each small partition in his inference space with relevant storage is called a "world." He stores problem solving methods and rules which he can handle at once in each world. He can retrieve and utilize these methods in a certain world, only when he makes inferences in the world. A contradiction can be found when he utilized two different worlds in solving problems. One is the world of well-formulated physics for Question 1 in which he stores the knowledge learned through the curriculum of physics, e.g., formulas and definitions, and another is his naive physical world for Question 3 which has been deeply engraved on his memory since his childhood, for instance. He has a "motion implies a force" misconception (Clement, 1982) in the naive one in this case. It is inconsistent with the knowledge for "uniformly accelerated motion" in the well-formulated one. He has answered

that the force is directed upward because he used the naively misconceptualized world.

"Student knowledge contradictions" are defined in this paper as the status which causes behaviors which can be regarded as a contradiction viewed from the standpoint of an observer. A typical interpretation of contradiction is as follows:

- He places more than two series of problem solving methods, which are originally placed in different worlds of concepts, in the same world regardless of their attributes. This is caused by his failure in differentiating them from each other.
- He makes decisions which are able to derive different truth values for a certain fact within a limited time, since his knowledge is unstable.

2.2 Formulations of student knowledge contradictions based on the multi-world logic

Here we discuss a formulation of student knowledge contradictions based on the concept discrimination structure and multi-world logic mentioned in Section 2.1. It is an intractable problem to try to model a student's inconsistent knowledge in a single reasoning space, since it cannot represent contradiction. This suggests we need another modeling paradigm which is able to cope with inconsistency.

The formulation is based on the authors' speculation that human beings partition their whole storage and inference space into multiple "worlds" and organize them in a discrimination tree to retrieve their knowledge efficiently by

1. first retrieving which world (concept) the given problem belongs to along a certain discriminating structure, and
2. retrieving and executing methods that contribute to problem solving in the world corresponding to the problem.

The first step, i.e., the decision on the target world, is regarded as a search on a *concept discrimination tree* as depicted in Figure 2. Each node of a discrimination tree corresponds to a concept and each leaf node to a world. When a problem is given to the current model, the discrimination tree in the model is at first interpreted to retrieve the conceptual world the problem belongs to by searching the tree from its root. The searching and retrieval process is called *discrimination level reasoning*. The second step is regarded as a search for and execution of methods that contribute to solving the problem. A set of problem solving methods is defined inside each world. *Method level reasoning* is thus carried out in the retrieved world, each of which consists of consistent knowledge. Contradiction is modeled as a mixture of multiple worlds which is caused by an incorrect discrimination tree structure. Roughly speaking, therefore, student knowledge contradictions are represented as wrong concept discrimination conditions, while conventional bugs are represented as wrong methods inside each world.

Problem solving knowledge is represented as a set of predicates whose formula is either `solve(`$G, \tilde{X}_{\rm in}, \tilde{X}_{\rm out}$`)` or `goal(`$\tilde{X}_{\rm in}, \tilde{X}_{\rm out}$`)`. G denotes the goal of the problem, that is, what should be determined under what constraints. $\tilde{X}_{\rm in}$ is a vector of input variables which are instantiated and $\tilde{X}_{\rm out}$ is a vector of output variables which are not instantiated when the predicate is called. $\{\tilde{X}_{\rm in}, \tilde{X}_{\rm out}\}$ represents whole the articulation of the problem space. For instance, the problem space "motion" is represented as $\{m, (s(t), A_s), (v(t), A_v), (a(t), A_a), (f(t), A_f), [(T_0, S_0, V_0, a_0, F_0), \cdots]\}$, where the elements are the mass of the moving object, location, velocity, acceleration and applied force as functions of time elapsed, and sets of the elements of the motion, respectively. Each function of time elapsed is denoted as a couple of the function itself and the attribute of the function. The problem space, which is adopted in Question 1 in Figure 1, is represented as $\{2,\ (s(t)),\ (v(t)),\ (a(t), fixed),\ (f(t)),\ [(2, S_0, (0,0), a_0, F_0),\ (0, (0,0), (19.6, 0), a_1, F_1)]\}$. When the problem solving begins, input variables are given in the formula as instantiated variables. For instance, the location and the velocity on $t = 0$ are instantiated as $(0, 0)$ and $(19.6, 0)$, because they are given in the problem. Problem solving is a retrieval and an execution of methods described in the problem solving knowledge base, to get the output parameter list $\{\tilde{X}_{\rm out}\}$ instantiated from the given input parameter list $\{\tilde{X}_{\rm in}\}$.

The given problem is represented as a vector of primitive attributes which is used for deciding the worlds the problem belongs to by tracing the given tree from its root. Each conceptual node on the tree that is not a leaf node has no problem solving methods but *discrimination knowledge*, which defines what

Figure 2: Concept discrimination tree

children nodes the node has and what conditions the problem must satisfy to discriminate each child concept from others. The conceptual node N_i that has children concepts $N_{i1} \cdots N_{ij}$ has discrimination knowledge $d(N_i, [N_{i1}, \cdots, N_{ij}], [C_{i1}, \cdots, C_{ij}])$ in general. To go forward through the path from N_i to N_{ik} $(1 \leq k \leq j)$ requires that $\{\tilde{X}_{\text{in}}\}$, input variables of the problem, should satisfy C_{ik}, the discrimination condition for N_{ik}. In Figure 2(a), the discrimination knowledge $d(N_1, [W_1, W_2], [C_1, C_2])$ is defined for the conceptual node N_1 that has children worlds W_1 and W_2, e.g., to go from N_1 to W_1 requires the problem to satisfy the condition C_1. Let us assume that N_1 corresponds to the concept of "linear motion", W_1 to "uniform motion", and W_2 to "linear and uniformly accelerated motion." Given the vector for Question 1 in Figure 1, it traces a path on the tree from the root and reaches the world W_2 via N_1, because it satisfies the discrimination condition C_2, e.g., $constant(A_t)$ which means "the acceleration is constant." In each world, $m(W_i)$, a set of problem solving methods which belong to W_i, is given. A method level student model, i.e., model of the method set $m(Wi)$, corresponds to a conventional student model. When S_w, the set of the worlds to which the given problem belongs, is identified, methods which contribute to solving the problem are retrieved and executed from the union of method sets, each of which is an element of S_w. When only one world W_2 is retrieved, method level reasoning takes place inside the world, i.e., problem solving methods are retrieved from $m(W_2)$ and executed. A discrimination condition has a predicate formula, which is called a *world predicate* and both C_1 and C_2 are world predicates in Figure 2. Each world predicate takes only input variables, i.e., $\{\tilde{X}_{\text{in}}\}$, as its argument.

Student knowledge contradictions are modeled in terms of erroneous concept discrimination trees, because the student who has such an erroneous tree cannot manage consistency in retrieving problem

Figure 3: Interpretation of the student contradiction

solving methods as mentioned above. Such kinds of contradiction can be modeled by erroneous world predicates in the multi-world model. When a student has not fully discriminated concepts N_{ip} and $N_{iq}(1 \leq p < q \leq j)$ which are both children nodes of N_i, the discrimination knowledge of N_i in the student model for him is revised to be $d(N_i, [N_{i1}, \cdots, N_{ip}, \cdots, N_{iq}, \cdots, N_{ij}], [C_{i1}, \cdots, C_{ip} \vee C_{iq}, \cdots, C_{ip} \vee C_{iq}, \cdots, C_{ij}])$, i.e., discrimination conditions for both N_{ip} and N_{iq} become $C_{ip} \vee C_{iq}$. Method level reasoning of the problem that belongs to any descendant worlds of either N_{ip} or N_{iq} takes place in the combined reasoning space $m(N_{iq} @ N_{iq})$ such that both method sets $m(N_{ip})$ and $m(N_{iq})$ are merged into it, where $m(N_{ip})$ contains all the methods defined in all the descendant worlds of N_{ip} and $m(N_{iq})$ contains those of N_{iq}. The model of the student who has not fully discriminated uniform motion and uniformly accelerated motion, in the above example, is represented by the discrimination knowledge of N_1, $d(N_1, [W_1, W_2], [C_1 \vee C_2, C_1 \vee C_2])$ as depicted in Figure 2(b). Method level reasoning is done in the combined reasoning space $m(W_1 @ W_2)$. Such model representation and interpretation enables the capture of his unstable applications of knowledge. When he becomes able to discriminate these concepts, the discrimination model is revised again to be correct and method sets that had been combined since then are restored.

Figure 3 illustrates a few instances of simplified student models. Each predicate written beside each arc of the tree, e.g., *solve_in_naive_world*, is the world predicate which discriminates the child conceptual nodes of the parent one. The student model for a student who has not yet discriminated between uniform motion and uniformly accelerated motion is illustrated in Figure 3(a). World predicates for these concepts are revised to be $uniform\_motion(\tilde{X}_{\rm in}) \vee uniformly\_accelerated\_motion(\tilde{X}_{\rm in})$. Figure 3(b) represents the status of the student who has the "motion implies a force" misconception in his naive world and unstably applies it in solving physics problems. World predicates for the naive world "$solve\_in\_naive\_world(\tilde{X}_{\rm in})$" and that for formulated worlds "$solve\_in\_formulated\_world(\tilde{X}_{\rm in})$" are both revised to be $solve\_in\_formulated\_world(\tilde{X}_{\rm in}) \vee solve\_in\_naive\_world(\tilde{X}_{\rm in})$. Although student's undifferentiation of concepts is represented by the disjunctive of world predicates as is explained above, conventional bugs are represented by wrong methods in a certain world. The student who has wrong knowledge, say knowledge for calculating location of the object that is moving linearly and uniformly, is modeled so that there exists at least a wrong method, e.g., "$S_t = V_0 * t$" in place of "$S_t = S_0 + V_0 * t$," in the world of uniform motion.

More detailed multi-world model descriptions are given in Figure 4 using the geographic domain. The domain is structured aiming to make a student understand that climates of seaside regions are

```
d(earth, [southern_hemisphere, northern_hemisphere],
    [southern_hemisphere(Place),
    northern_hemisphere(Place)]).
m(southern_hemisphere, [
    (grow(Crop, Place, T) :-
        suitable_temperature(Crop, Place, T1),
        suitable_soil(Crop, Place, T2),
        suitable_lay(Crop, Place, T3),
        has_irrigation(Place, T4),
        and(T1,T2,T3,T4,T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_latitude(Place, T))]).
m(northern_hemisphere, [
    (grow(Crop, Place, T) :-
        suitable_temperature(Crop, Place, T1),
        suitable_soil(Crop, Place, T2),
        suitable_lay(Crop, Place, T3),
        has_irrigation(Place, T4),
        and(T1,T2,T3,T4,T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_latitude(Place, T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_high_latitude(Place, T1),
        west_coast(Place, T2),
        and(T1,T2,T))]).
```

(a)

```
d(earth, [southern_hemisphere, northern_hemisphere],
    [southern_hemisphere(Place)∨northern_hemisphere(Place),
    southern_hemisphere(Place)∨northern_hemisphere(Place)]).
m(southern_hemisphere, [
    (grow(Crop, Place, T) :-
        suitable_temperature(Crop, Place, T1),
        suitable_soil(Crop, Place, T2),
        suitable_lay(Crop, Place, T3),
        has_irrigation(Place, T4),
        and(T1,T2,T3,T4,T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_latitude(Place, T))]).
m(northern_hemisphere, [
    (grow(Crop, Place, T) :-
        suitable_temperature(Crop, Place, T1),
        suitable_soil(Crop, Place, T2),
        suitable_lay(Crop, Place, T3),
        has_irrigation(Place, T4),
        and(T1,T2,T3,T4,T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_latitude(Place, T)),
    (suitable_temperature(Crop, Place, T) :-
        middle_high_latitude(Place, T1),
        west_coast(Place, T2),
        and(T1,T2,T))]).
```

(b)

Figure 4: Examples of multi-world student model representation in Prolog

different between the Southern hemisphere and the Northern hemisphere because of the effects of sea currents. The whole correct model except knowledge of instances (correct facts) is given in Figure 4(a). The conceptual node "earth" is the root of the discrimination tree and has two children worlds "southern_hemisphere" and "northern_hemisphere." World predicates "`southern_hemisphere(Place)`" and "`northern_hemisphere(Place)`" are given for discrimination conditions. Each input variable, i.e., `Place`, is bound to a name of the region and the discrimination knowledge evaluates which world the region belongs to. If a region is in the Northern hemisphere, world predicate `northern_hemisphere(Place)` indicates true and world northern_hemisphere is selected. Method level student models are actually represented in SMDL which is an extended version of Prolog and is formulated later, though they are written in Prolog in the figure. In each clause of method level models, only the last parameter, the truth variable, is the output variable and all other parameters, i.e., `Crop` and `Place`, are input variables. The correct method level student model of the world southern_hemisphere consists of two clauses, which mean "The `Crop` grows in a certain `Place`, if the temperature, the soil, and the lay of the `Place` is suitable and there exists enough irrigation," and "The temperature of a certain `Place` is suitable for the `Crop`, if the `Place` is the middle latitudes," respectively. The correct method level model of the world northern_hemisphere is additionally given a clause that means "The temperature of a certain `Place` is suitable for the `Crop` even if the `Place` is in relatively high latitude, if the `Place` is on the west cost of a continent," because warm currents and wind from the west bring such regions a warm climate in the Northern hemisphere.

The discrimination knowledge of the student model in Figure 4(b) is revised from the correct one (underlined), i.e., both world predicates for southern_hemisphere world and northern_hemisphere world are `southern_hemisphere(Place)`∨`northern_hemisphere(Place)`. Such a representation of the discrimination knowledge denotes that the student has not yet correctly discriminated the two worlds. Given a problem on this subject, for instance, "Does wheat grow in the southern part of Chile?," both of the discrimination conditions indicate true, so the student model interpreter generates a combined method set of the two worlds. Then the model in the southern_hemisphere world correctly returns *false*, but the model in the northern_hemisphere world returns *true* which causes a contradiction.

Table 1: Classification of contradictions in student modeling

(a) Classification by cause

Classified name	Type
Student Contradiction	[A1][A2][A3]
Modeling Contradiction	[B]

(b) Classification by handling

Classified name	Type
Single World Contradiction	[A1][A2][B]
Multi-World Contradiction	[A3]

2.3 Classification and handling of contradictions

Contradictions enumerated at the beginning of this section are classified by their causes into two categories:

1. *student contradictions* ([A1][A2][A3]),
2. *modeling contradictions* ([B]).

Student contradictions are caused by inconsistencies of a student's behavior or his knowledge itself. On the other hand, modeling contradictions are caused by inconsistencies between the knowledge that a student actually has and the knowledge that is represented in the student model. This classification is summarized in Table 1(a).

Next, let us classify these contradictions from the viewpoint of how they should be handled in student modeling. This classification is summarized in Table 1(b).

An inconsistency between a student's actual understanding and a student model must be caused by faulty assumptions that were hypothesized in a previous inference stage and have been believed. A nonmonotonic student modeling process is required to get rid of these inconsistencies, i.e., to revise the student model so as to be consistent with the student's knowledge:

1. set up assumptions that are necessary to construct the student model which satisfies the set of given oracles in each phase of student modeling,
2. derive the model from these assumptions and record the derivation process,
3. when a certain inconsistency between the oracle set and the model is detected, find a set of assumptions which causes the inconsistency,
4. resolve the contradiction by revising the system-made assumptions in the set and continue the modeling.

Considering each oracle as an assumption, the above nonmonotonic modeling methodology to resolve the inconsistencies is applicable to contradictions of types [A1] and [A2]. Oracles that are not consistent with a student's *current* understanding should be removed from the current assumption (oracle) set. A student model that is consistent with a student's understanding can be constructed from the assumption set manipulated in such a way inside a single world. *Single world contradiction* is a general term for these types of contradictions, i.e., [A1], [A2] and [B], which a modeling system should try to build a consistent model of in a single world by finding an appropriate set of oracles (assumptions).

The above four steps of the belief revision process suggest that ATMS is appropriate for a core module of the controlling mechanism for student modeling. In ATMS-based problem solving systems, ATMS and an inference system work in collaboration with each other. The inference system executes problem solving and informs ATMS of its inference process. The consistency among data dealt with by the inference system are managed by ATMS. ATMS holds and revises a set of valid assumptions which is the origin of the inference and derivation process of data by the inference system. When derivation of a contradiction is informed, ATMS calculates the set of assumptions which causes the contradiction by tracing back the informed derivation paths from the contradiction. When an assumption is denied, data which rely on it can not hold any more and hence are automatically denied by ATMS. In addition,

ATMS is able to avoid redundant calculations which have previously been made, which is useful to make the inference process very efficient.

All the single world contradictions, i.e., [A1], [A2] and [B], are dealt with by the same mechanism, which is realized by formulating the inductive student modeling process on the basis of ATMS in HSMIS. A detailed description of this topic is given in Section 3.

In the case of student knowledge contradictions, i.e., type [A3], on the other hand, contradictions in his knowledge should not be resolved but should be represented as they are. As is discussed in the above two subsections, such a type of contradiction is represented well on the basis of multi-world logic, and is called a *"multi-world contradiction"* in contrast with a single world contradiction. Based on the multi-world formulation, a student knowledge contradiction can also be defined as a certain kind of inconsistency which arises among some assumptions that have been assumed in the inference process in a manner similar to a single world contradiction. It is hence possible to capture a student's contradictory knowledge by formulating its detecting/handling methodology.

The student modeling system THEMIS which is able to cope with multi-world contradictions consists of HSMIS and the controlling mechanism of multiple worlds. HSMIS makes inferences consistently in each world coping with and resolving single world contradictions. When a multi-world contradiction is detected, HSMIS passes the control to the Multi-World Controller. It revises the concept discrimination tree of the given domain knowledge to represent the contradiction. A detailed description of this topic is given in Section 4.

2.4 Heuristics to distinguish contradictions

It is difficult for not only modeling systems but also human teachers to distinguish and detect the four types of contradictions, i.e., type [A1], [A2], [A3] and [B], which are the essentials in student modeling as already mentioned, because all of their indications are very similar (Dillenbourg *et al*, 1992; Self, 1993b). They are triggered by a difference between the expectation of the student answer deduced from the current student model and his actual answer. One of the research goals of this paper is to produce a generic and formulated modeling mechanism which is able to cope with these kinds of contradictions. Although a generic methodology to distinguish them is not fully developed, some heuristics are employed as shown below.

Let us assume that the certainty of every given oracle and clause in the student model can be available. Both single world contradictions and multi-world contradictions are detectable by quite similar triggers, i.e., the expectation from the model and the actual oracles. The contradiction resolving procedures of those contradictions are quite different from each other. Single world contradictions need to be resolved by revising the set of oracles or the current model in general. The contradiction resolution procedure for each type of single world contradiction is a bit different, and hence the detection processes for them are different from each other. In the heuristics, multi-world contradictions are first distinguished from single world contradictions.

Student knowledge contradictions, i.e., type [A3], should not be resolved, because the student's inconsistency should be modeled as he is. Student knowledge contradictions require the revision of neither the oracle set nor the clauses that are inconsistent with oracles, but the discrimination structure to permit it to contain the inconsistency. Such a difference in treatment of multi-world contradictions and single world contradictions suggests the following way of discriminating them. If either the certainty of a clause which is inconsistent with valid oracles or certainties of some of the oracles are less than a certain threshold, the inconsistency should be considered to be a single world contradiction and hence should be resolved. On the other hand, if all the certainties are high enough, the inconsistency is considered to be a student knowledge contradiction. They are not revised but put into some worlds, i.e., all the reliable data can be alive in the multi-world formulation. The following heuristics to detect contradictions of each sub category of single world contradictions are incorporated.

The change of student's knowledge which causes type [A1] of contradictions occurs especially right after his errors are corrected. He then generally changes his understanding from an erroneous to a correct status. It is appropriate to apply revision procedures for type [A1], when correct oracles are obtained right after tutoring, i.e., the system resolves the contradiction by excluding the past oracles inconsistent with correct clauses, or by asking him truth values of the oracles. The revision of

oracles results in the revision of the model, i.e., erroneous clauses are dismissed and correct clauses are appended.

Independently of the correctness, generally speaking, the student ought to have consistently applied the clauses that are inconsistent with newly obtained oracles throughout a certain period, in the case that he makes careless mistakes which cause type [A2] of student contradictions. Thus such a type of contradiction is detected by similar criteria as those for student knowledge contradictions, i.e., the inconsistent oracles and clauses would be both reliable enough. There are two ways to distinguish them; one is to consider a situation as a type [A2] only when the situation could not be treated as a student knowledge contradiction, and another is to ask him a very similar question to get a confirmation.

These contradictions can be more sufficiently distinguished by introducing and enriching domain dependent heuristics, e.g., "Students tend to mistake a uniformly accelerated motion for a uniform motion if the motion is vertical," in addition to the domain independent heuristics explained above.

There is one more point which should be considered in designing a student modeling system. It can be assumed that there exists a student who hardly behaves consistently, because of his low ability or the system's inappropriate selection of the level of task. It does not make sense to let such a student complete the current task. It is possible to detect such a status of the student by diagnosing the past record of acquired oracles. In such cases, the modeling system should give up modeling him and inform the monitor of the failure so as to let the student go back to elementary tasks.

3 Nonmonotonic student modeling in a single world

On the basis of the above conceptual-level discussion on contradictions in student modeling, this section proposes a powerful nonmonotonic inductive student modeling methodology which is able to cope with single world contradictions. Student models have to satisfy the following requirements in addition to the requirements for the nonmonotonicities already discussed:

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 a tradeoff between the accuracy of the model and the cost of constructing it. From a pragmatic viewpoint, we must set up an appropriate representation scheme for student models by taking the tradeoff into consideration.

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

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

To meet these requirements, the authors 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 (Model Inference System) (Shapiro, 1982; Shapiro, 1981), is an inductive inference system for SMDL. In HSMIS, ATMS: Assumption-based Truth Maintenance System (deKleer, 1986) is employed for dealing with nonmonotonicities. HSMIS has been implemented in Common ESP(Extended Self-contained Prolog) on SPARC station (AIR, 1990).

3.1 SMDL: A student model description language

In addition to the above requirements, a student model is required to represent not only students but also the systems' 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, to denote the above two states, respectively. Needless to say, the former state, i.e., the 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

Table 2: 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

ascertain its truth", respectively. Then, we have two seemingly the same truth values "false", which can be discriminated as follows: Employing Prolog terminology, the former "false" is treated as "fail" and the latter as one of the three values corresponding to "success." Discrimination among the three values is done by introducing an auxiliary argument interpreted by a meta-interpreter.

Facts are represented in SMDL as follows.

`torrid(paris,false).` `temperate(paris,true).` `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."

Clauses are written in the form of

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

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

`grow(`X`,`T_4`) ::- temperate(`X`,`T_1`).`

`grow(`X`,`T_5`) ::- torrid(`X`,`T_2`), wet(`X`,`T_3`).`

These two clauses show that the student thinks "If place X is in the temperate zone or in the torrid and 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`)` , the SMDL interpreter calls the subgoals `temperate(paris,`T_1`)` , `torrid(paris,`T_2`)` and `wet(paris,`T_3`)` in this order. The truth value T of `grow(paris,`T`)` is obtained according to $T = T_4 \vee T_5 = T_1 \vee (T_2 \wedge T_3) = true \vee (false \wedge unknown)$. The semantics of the logical operators "$\wedge$" and "$\vee$" are shown in Table 2.

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 variable or one of four truth values.

An SMDL clause is of the form: H::-$B_1, B_2, \cdots, B_n$, $n \geq 0$, where H, B_i $(1 \leq i \leq n)$ are predicates. In the case of $n = 0$, 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-resolution and strong-resolution. The former is like the execution process of Prolog, i.e., "a goal succeeds in weak-resolution iff there exists at least one clause which derives the goal" (see Definition A.1 in Appendix A). On the other hand, strong-resolution is somewhat different and complicated. Roughly speaking, a goal with truth value T succeeds in strong-resolution, iff all the OR clauses unifiable with the goal succeed and the result of the OR evaluation is T (see Definition A.2 in Appendix A). Generally speaking, a goal with truth value *true* can be weakly derived but a goal with another truth value needs to be strongly derived. For instance, a goal `grow(paris,`T`)` succeeds in weak-resolution with truth value *true* by executing only the first clause, if a subgoal `temperate(paris,`*true*`)` succeeds. On the other hand, a goal `grow(antarctica,`T`)` needs to succeed in strong-resolution with truth value *false* by executing both clauses to call the subgoals `temperate(antarctica,`T_1`)` , `torrid(antarctica,`T_2`)` and `wet(antarctica,`T_3`)` , whose execution results are *false*, *false* and *true*, respectively, and to obtain the truth value of the goal according to $false \vee (true \wedge false)$. Formal definition of the execution of goal G with program P is given in Definition A.3.

Figure 5: Block diagram of THEMIS.

3.2 SMIS: Inductive student model inference system

Figure 5 shows the block diagram of THEMIS. HSMIS, the core module of THEMIS, consists of SMIS (Ikeda *et al*, 1989), ATMS (explained below), The Virtual oracle generator (also explained below) and the Contradiction resolving system (CRS). The main task of ATMS is to manage the consistency of a set of assumptions (environment) used by the problem solver, SMIS in our case. The 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 the student questions. CRS resolves the inconsistency identified by changing the environment.

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. 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 $T' \in \{ true, false, unknown \}$ (see Definition B.1 in Appendix B). Because $fail$ represents the system's understanding of the 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.

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 Ω-refuted clause or addition of a new clause, and informs ATMS of the process.

3.2.1 Student model diagnosis system

SMDS traces the resolution process of the current model and checks the results with oracles. SMDS has three subprocedures, that is, *ip* , *fp* and *failp* . The procedure *ip* finds out where a new clause should be added. The procedure *fp* detects an Ω-refuted 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(See Table 3).

Let us assume that there exists an oracle $H =< p(\tilde{X}', T), T' >$, a clause $C = p(\tilde{X}, T)$::-$q_1(\tilde{X}_1, T_1)$, $q_2(\tilde{X}_2, T_2)$, $\cdots$, $q_k(\tilde{X}_k, T_k)$, and the set of oracles $q_1(\tilde{X}'_1, T'_1), q_2(\tilde{X}'_2, T'_2), \cdots, q_k(\tilde{X}'_k, T'_k)$ that derives $p(\tilde{X}', T')$ with C. The set of oracles is called a *top-level-trace* of C for H (see Definition B.2 in Appendix B). A simplified but concrete example of a top-level trace is shown in Figure 6, where the

Table 3: Activation of SMDS subprocedures

Figure 6: Examples of the top-level trace and the refutation for a clause.

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.

Let us assume that there exists an oracle $O_7 : <$ `grow`$(\texttt{rice}, \texttt{kiev}, T_7), \texttt{true} >$. The top-level trace of C for O_7 can be made, iff there exist the following two oracles:

$O_8 : <$ `suitable_temperature`$(\texttt{rice}, \texttt{kiev}, T_8), \texttt{true} >$ and

$O_9 : <$ `suitable_soil`$(\texttt{rice}, \texttt{kiev}, T_9), \texttt{true} >$.

However, when there exists an oracle $O_8' : <$ `suitable_temperature`$(\texttt{rice}, \texttt{kiev}, T_8'), \texttt{false} >$ instead of O_8 and there are no OR clauses unifiable with the goal O_7, O_7 cannot be weakly-derived from the current model. In such a situation, the model P is said to be *too weak* with respect to a goal O_7 (see Definition B.3 in Appendix B). When a weakness is detected in P, *ip* is activated and it searches for the cause of the weakness by proving each clause in the proof tree of the oracle. Formal definition of *top-level-trace* and *weakness* is given in Appendix B.

Figure 7 shows a Prolog implementation of *ip* . The goal good_oracle_top($p(\tilde{X}, T)$, T', Body, T_b) finds a clause $p(\tilde{X}, T)$::- Body $\in P$ which has a correct top-level trace. If such a clause does not exist, the goal $p(\tilde{X}, T')$is uncovered by P and returned as an output by *ip* . Otherwise, *ip* is recursively called.

To cover the uncovered goal detected by *ip* , HSMIS searches for a new clause to add into the model.

We define a binary relation "$\succ$" over truth values. When two clauses derive different truth values, T_1 and T_2, for a goal, we say T_1 is *stronger* than T_2 iff $T_1 = T_1 \vee T_2$ and write $T_1 \succ T_2$ (see Definition B.4 in Appendix B).

We say a model is *too strong* with respect to a goal, if it has at least one clause which derives a stronger truth value than the oracle with respect to the goal (see Definition B.5 in Appendix B). A too strong model has at least one Ω-refuted clause which has a refutation (see Definition B.6 in Appendix B). In Figure 6, the clause C is refuted by oracles O_4 , O_5 and O_6. The Ω-refuted clause, which should

```
ip( (A,B), (Ta, Tb), C ) :-
       weakly_derive( A, Ta)
         →  ip( B, Tb, C );
            ip( A, Ta, C ).
ip( A, Ta, C ) :-
       good_oracle_top( A, Ta, Body, Tb)
         →  ip( Body, Tb, C );
            C = [A::Ta, uncover].
```

Figure 7: Prolog implementation of ip

```
fp( A, T, C ) :-
       refutation( A, T, Body, Tb),
       check_refutation( Body, Tb, Cb),
       ( Cb==ok
         ->  C = [A::-Body, incorrect];
             C = Cb).
check_refutation( (A,B), (Ta,Tb), C ) :-
       check_refutation( A, Ta, Ca)
       ( Ca==ok
         ->  check_refutation( B, Tb, C );
             C = Ca).
check_refutation( A, Ta, C ) :-
       oracle( A, T )
       ( ge( T, Ta)
         ->  C = ok ; fp( A, T, C ) ).
```

Figure 8: Prolog implementation of fp

be removed from the model as a cause of *strength*, is identified by the procedure *fp* .

Figure 8 shows a Prolog implementation of fp. The goal refutation($p(\tilde{X}', T)$, T', Body, T_b) finds the clause $p(\tilde{X}, T)$::- Body $\in P$ which weakly-derives $p(\tilde{X}', Tg)$ such that $Tg \succ T'$. The goal check_refutation(Body, Ta, C) checks each goal of the Body with Ω .

Let us assume that check_refutation($q_i(\tilde{X}_i', T_i)$, T_i', C) is activated under the condition that $q_i(\tilde{X}_i', T_i')$ is derived ¿from P and $< q_i(\tilde{X}_i', T_i), T_i'' >$ is in Ω. If $T_i'' \prec T_i'$ then the cause of *strength* should be found in the resolution process of $q_i(\tilde{X}_i', T_i')$. Therefore check_refutation calls fp($q_i(\tilde{X}_i', T_i)$, T_i'', C) recursively. Otherwise it returns C=ok.

The model P is said to be *incomplete*, if P derives the truth value $fail$ to a certain fact which is contained in Ω. The procedure *failp* identifies the cause of the *incompleteness*, which is either an uncovered goal or an Ω-refuted clause. Figure 9 shows a Prolog implementation of failp. The procedure failp($p(\tilde{X}', T)$, T', C) is activated when the model P cannot derive $p(\tilde{X}', T')$ and $< p(\tilde{X}', T), T' >$ is in Ω. If P cannot weakly-derive $p(\tilde{X}', T')$, it calls ip($p(\tilde{X}', T)$, T', C). Otherwise, it calls fp2($p(\tilde{X}', T)$, T', C). The procedure fp2 selects the clause which derives $p(\tilde{X}', fail)$ and finds a correct top-level trace of the clause. If the correct top-level trace derives $p(\tilde{X}', Tg)$ and $Tg \succ T'$ then fp2 returns the clause as an Ω-refuted clause. Otherwise fp2 calls *failp* for the body of the clause. The formal definition of *incompleteness* is given in Definition B.7.

3.2.2 Search for a new clause

A candidate clause to be added into the model is generated using the refinement graph which is defined by refinement operators. The generation process is viewed as a kind of search on a dynamically generated tree. The refinement graph is rooted by a most general clause. Its nodes correspond to candidate

```
failp( (A,B), (Ta,Tb), C ) :-
       smdl( A, fail )
         ->  failp( A, Ta, C );
             failp( B, Tb, C ).
failp( A, Ta, C ) :-
       not( weakly_derive( A, Ta) )
         ->  ip( A, Ta, C );
             fp2( B, Tb, C ).
fp2( A, Ta, C ) :-
       smdl_clause( A, Body ),
       smdl( Body, fail ),
       good_oracle_top( A, Tg, Body, Tb),
       ( le( Tg, Ta), failp( Body, Tb, C );
       ( gt( Tg, Ta), C = [A::-Body,incorrect] )).
```

Figure 9: Prolog implementation of failp

clauses and directed arcs $C \xrightarrow{\rho} C'$ to the refinement operations applied to clauses. Typical examples of refinement operations include adding a predicate into the body and unification of variables. The graph has the following important characteristic:

If the clause C does not cover a goal A, then any C' satisfying $C \xrightarrow{\rho} C'$ does not cover it either.

This is used for pruning the unnecessary branches, which makes the search efficient.

A refinement graph represents knowledge which enables an 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 a clause from a fixed root, i.e., the most general clause, independently of material. 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, the search procedure can begin searching from these bugs, which makes the search very efficient.

3.3 ATMS

This section briefly explains ATMS. Detailed description is given in Appendix C.

The information given by the inference system takes a form of

$$N_1, N_2, \cdots, N_k \Rightarrow D$$

which means that the datum D is derived from a set of the data $\{N_1, N_2, ..., N_k\}$. $\{N_1, N_2, ..., N_k\}$ is called a justification of D .

The data dealt with in the inference system are classified into three kinds of data, i.e., premise data, assumed data and derived data. A premise is defined as a datum that can be true under any context. An assumed datum is one produced with an assumption that holds without depending on any other data. A derived datum is one inferred from other data.

Following each justification back from a certain derived datum finally reaches a set of assumptions and premises. That is to say, the set of assumptions that an individual datum depends on can be calculated. A set of assumptions is called an environment. It is one of the major tasks for ATMS to record justifications informed from the inference system and to calculate a consistent environment where the data can be inferred. When derivation of the contradiction is informed, ATMS calculates the nogood environment, which is the cause of the contradiction and recorded in ATMS (hereafter called the nogood record). Every environment included in the nogood record can be regarded as an inadequate combination of the assumptions. ATMS maintains the consistency of the inference process by using the nogood record. The inference system selects a new consistent environment, which does not include the nogood record elements, and continues inference.

A situation in the problem solving process is called a context, which is defined by the set of the

data that hold in the situation. An environment deriving all the data included in the context is called a characteristic environment of the context.

When the derivation of an inconsistency is informed, ATMS calculates and records the environment of $\perp$. The inference system ceases to solve the problem in the contradicted context and transfers to a new consistent characteristic environment.

With regard to the nodes which have been derived before that time, ATMS determines whether each node holds (*in*) or does not hold (*out*) in the new characteristic environment. Thus a new context is composed with a set of *in* nodes.

3.4 Managing consistency to control the model building process

Single world contradictions are formulated in a unified architecture in HSMIS by combining SMIS and ATMS. With the aid of the schematic diagram shown in Figure 5, the overall behavior of the system for single world contradictions will be made clear in this subsection. (1) Given student answers ("real oracle"), the Virtual oracle generator generates virtual oracles if necessary and passes them to ATMS with the real oracles. (2) SMIS informs ATMS of all the inference process that is explained in detail in Appendix D. When a contradiction is informed, ATMS computes the label responsible for the inconsistency based on the information given up to that point of time and stores 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) passes the control to SMIS together with a new context supported by the new consistent environment.

HSMIS tries to model the student from his behavior during which it automatically asks questions which contribute to the 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. The following additional mechanisms are 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 she assumes that he usually understands the material very well. Although the former case is reasonable, the system would tend to ask many questions to get sufficient information about how well he 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 characteristic is very reasonable in real tutoring. The latter is employed. A serious problem still remains, however. One cannot simply put a clause into the student model without any justification.

The Virtual oracle generator, which generates plausible student answers based on the certainty of the current student model instead of asking questions, has been devised in order to cope with this problem. When a student's behavior is confined within the scope of his teacher's prediction, she asks fewer questions by replacing the necessary information with correct answers. This type of oracle is called a "*virtual oracle*." SMIS treats "real oracles" and "virtual oracles" in the same manner, while ATMS manages their consistency. Detailed formulation of virtual oracle is given in Appendix D.2.1.

When a clause supported by virtual oracles turns out to be no longer *in* as the inference proceeds, ATMS withdraws it and backs up to the point which causes the problem.

Meta-Oracles: Students sometimes want to express their knowledge in the form of knowledge instead of facts. The system sometimes wants to ask the student the reason why he 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 fields and rivers.
```

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

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

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

The clause obtained from the student is called a "*meta-oracle.*" The formulation is given in Appendix D.2.2.

3.5 Assumptions to cope with single world contradictions

This subsection gives the formulation of assumptions to drive the mechanism mentioned in Section 3.4. The nonmonotonic inference process of HSMIS is realized by controlling the status of the assumptions representing various hypothetical decisions which are made during student modeling. Detailed formulation of the assumptions and the inference process derived from them are explained in Appendix D.

Given a consistent oracle set $\tilde{\Omega}$, SMIS is able to construct a student model which explains $\tilde{\Omega}$. That means SMIS is potentially able to cope with all the modeling contradictions in terms of SMDL, if an appropriate $\tilde{\Omega}$ is given. In HSMIS, these modeling contradictions are formulated to be controlled by the following three types of assumption in the framework of ATMS:

uncover(C, O) which represents that an oracle O can not be covered by a certain clause C. It holds unless $cover(C, O)$ is derived. This type of assumption is used to prune branches in the refinement graph and to detect that the search turns out a failure.

Ω-consistent(C) which represents that a clause C is consistent with $\tilde{\Omega}$, the current oracle set. It holds unless $\Omega\text{-}refutation(C)$ is derived. Such a consistency is checked whenever $\tilde{\Omega}$ is changed by the mechanism. This type of assumption is used for revising the student model to follow change of $\tilde{\Omega}$.

general(C) which represents that any ancestor clauses of C in the refinement graph which are more general than C are not contained in the current model. It holds unless an ancestor clause of C is added to the model. If an ancestor clause of C is added to the model, the clause C no longer holds in the model.

In order to cope with two types of student contradictions, [A1] and [A2], which are both classified into the single world contradiction, it is necessary to update the oracle set itself, e.g., to leave unreliable oracles out of consideration in modeling or to modify truth values of some of the oracles which cause a certain inconsistency. This requirement means that the system should be able to generate the currently reliable set of oracles $\tilde{\Omega}$, which is considered in model inference, from the whole set of given oracles Ω. The following type of assumption is formulated in addition to the above three types to meet the requirement.

oracle(O) which represents a question and a student's answer to it, i.e., a "real" oracle O. If the assumption **oracle**(O) is *in* , oracle O is in $\tilde{\Omega}$. This means the oracle is reliable at present.

A modeling system must be able to realize more appropriate behavior if it copes with nonmonotonicities of a teacher's trust in student's knowledge and asks reasonable numbers of questions in the didactic sense. Contradiction of trust is formulated by incorporating the following assumption:

trust(C) which represents that the system trusts the student to have the knowledge that corresponds to clause C. It is *in* unless $\Omega\text{-}refutation(C)$ is derived. Virtual oracles are generated based on this assumption and the oracle that is unifiable with C. The number of system questions is hence reduced without losing logical consistency of the modeling process. When this assumption becomes *out* , virtual oracles which were derived from it are automatically withdrawn, and the system asks the student the facts which had been trusted and have not been asked until then.

SMIS deals with oracles and virtual oracles in the same manner.

3.6 Detection and resolution of contradiction

The contradiction derived in the inference process of HSMIS is classified into the following seven types.

S1) Contradiction of Ω-consistency
S2) Contradiction of cover test
S3) Contradiction of generality
S4) Contradiction of trust
S5) Contradiction of meta-oracle
S6) Contradiction of oracle
S7) Failure of search

When any of the contradictions is detected, ATMS is informed of it and updates the nogood record.

Among the assumptions which are formulated in HSMIS to cope with single world contradictions, Ω-**consistent**(C) , **uncover**(C, A), **general**(C), **trust**(C) and **metaOracle**(C) are called *default assumptions*, which means "they are assumed to be *in* so long as no contrary evidence is found." The set of these assumptions included in the environment is called a *default environment* (denoted by D_e). From this, the current environment C_e can be expressed by

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

Contradictions can be classified into (a) ones regarding the default environment and (b) ones regarding the oracle environment (corresponding to $\tilde{\Omega}$).

(a) The resolution method for contradictions of the default environment can be easily derived from its definition.
Suppose that a contradiction is detected since there exist both the Ω-$refutation(C)$ and the Ω-**consistent**(C) in the current context. As has already been stated, the Ω-**consistent**(C) can hold so long as there does not exist the refutation for the clause C ,i.e., Ω-$refutation(C)$. Therefore, the inconsistency can be resolved by removing the Ω-**consistent**(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.

(b) CRS generates consistent $\tilde{\Omega}$. The consistency is guaranteed to the extent that it does not include any contradictions that have been found thus far. In other words, it may contain a contradiction found in the future. The operations carried out for the generation are:

(1) remove an oracle from Ω,
(2) modify the truth value of an oracle in Ω.

CRS searches for a consistent $\tilde{\Omega}$ in the ascending order of number of modifications from Ω. To improve the efficiency and educational validity of selecting the consistent oracle set which has the minimal modifications, some domain-independent heuristics are incorporated into the search procedure: Give priority to correct or recent answers, give priority to the oracles supporting the plausible clause which is supported by relatively many oracles, and so on. Domain-dependent heuristics may also be introduced.

3.7 Behavior of HSMIS

An example of how HSMIS works is given in Figure 10, in which the description is partially simplified because of space limitation, i.e., the description of teaching material is not correct in the strict sense (for example, the first argument of grow is omitted). An example of the modeling process for a student knowledge contradiction is shown in Section 4.

In order to realize the above behavior of HSMIS, at least two kinds of knowledge are 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, i.e., $temp(Place, T_1)$ 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.

Figure 10: An example of HSMIS modeling process

The other kind is the correct domain knowledge which provides HSMIS with correct answers. In the above example, the knowledge is used for three purposes, i.e., for generating virtual oracles, managing the environment, and generating problems.

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 in the inference process, i.e., $C1$ 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 Nonmonotonic student modeling in multiple worlds

4.1 Inference process in THEMIS

This subsection discusses both the modeling mechanism and the formulation of the modeling process realized in THEMIS which consists of HSMIS and Multi-World Controller (MWC). MWC uses HSMIS as a single world modeler capable of resolving single world contradictions. THEMIS is hence able to represent student knowledge contradictions according to the multi-world logic, maintaining the consistencies between model and oracles in each world. A formulation of the inference process is given in detail in Appendix D.

Among student contradictions, the formulation of student knowledge contradictions is more complicated than formulations of the other two. The following two types of assumption are introduced to formulate student knowledge contradictions:

belong(P, W_i) which represents that the student is assumed to understand that problem P belongs to world W_i.

discriminate(W_i, W_j) which represents that the student is assumed to be able to correctly discriminate between the two worlds W_i and W_j.

When a correct assumption **belong**(P, W_i) is *in* , the student is expected to be able to correctly apply a problem solving method , say, $\mathcal{M}_i$, which is in the world W_i of the current model. If he does not discriminate between W_i and W_j, however, it can happen that he applies $\mathcal{M}_j$ which belongs to W_j to the problem. A contradiction is derived from the assumptions **belong**(P, W_i) which is pre-assumed and **belong**(P, W_j) which is newly assumed to cover his faulty answer. Another contradiction is also derived from the assumption **discriminate**(W_i, W_j) and the datum $indiscriminate(W_i, W_j)$ which is derived from assumption **belong**(P, W_j). In this case, oracles related to P are transferred from W_i to W_j together with the belief revision of **discriminate**(W_i, W_j) to be *out* and the system revises the discrimination conditions which discriminate the two worlds so as to resolve these contradictions. Thus the revised student model represents the student knowledge contradiction caused by his undifferentiation.

Contradictions derived in the inference process of THEMIS are classified into the following three types in addition to the contradictions formulated in HSMIS.

M1) Contradiction of belong
M2) Contradiction of discrimination
M3) Contradiction of model prediction and oracle

All these contradictions handled by MWC are resolved in a similar manner to contradictions of the default environment, i.e., "the assumptions which were hypothesized in advance and which derived the faulty expectation is denied to be *out* , when a contradiction is informed." The assumption **belong** which has been assumed becomes *out* in the case of type M1 contradictions. The assumption **discriminate** and the assumption **belong** which has derived an expectation of a student answer also become *out* in the case of type M2 and M3, respectively.

4.2 Modeling student knowledge contradictions

The formulation of student knowledge contradictions described earlier works well as a student modeling method by utilizing the heuristics which are also mentioned earlier. MWC is incorporated into THEMIS to control multi-world inference. Concept discrimination trees are given in advance as a part of the domain dependent knowledge. MWC is given the whole set of worlds which are handled in one course of tutoring, and it manages the status of each discrimination condition in the tree and each set of oracles that belong to each world. MWC is able to retrieve all the clauses in a certain world which are unifiable with a certain oracle in the world with the help of ATMS. Model diagnoses and revisions can be done on the discrimination trees. In each world, the clause level student model is inductively inferred ¿from the oracles belonging to the world. It is realized by modifying the algorithm of the SMDL interpreter, i.e., a clause C in a certain world W is unifiable only with oracles belonging to W. Each clause level student model can be consistently inferred using such a mechanism.

The construction process of the student model that represents student knowledge contradictions is as follows:

1. The system assumes a student knowledge contradiction when newly obtained oracles are not satisfied by any unifiable clauses in the corresponding world in the model which are sufficiently reliable.
2. The system tests whether the oracles are satisfied by the clauses that *exist* in another world by visiting the worlds in turn in order of similarity to the correct world according to the structure of the tree.
3. When a clause explaining the oracle is found in some world, discrimination conditions that contribute to differentiation of the two worlds are revised as depicted in Figure 3.
4. If no satisfiable world is found, the system considers the situation as a single world contradiction and tries to revise the model in the correct world.

Such a decision is made based on the heuristics to detect a student knowledge contradiction mentioned in Section 2.2. THEMIS calculates the certainty of each clause in its clause (method) level student model so that it is able to apply the heuristics. The calculation is done by referring to various kinds of information, i.e., number of top-level traces that justify the clause, whether the oracles which consist of each top-level trace of the clause are correct answers or not, how old the oracles are, etc (Kono *et al*, 1993a).

I_W, the set of instances whose every element originally belongs to a certain conceptual world W, can be determined by applying C_W which is the world predicate of W to I, the whole set of instances in the domain. Clause level representations and oracles which justify the model are generated and stored in each world individually. Suppose that there are two worlds W_1 and W_2 which are brothers and that they have already obtained and involved oracle sets $\tilde{O}_{W1}$ and $\tilde{O}_{W2}$. Suppose also that clauses $\mathcal{M}_1$ and $\mathcal{M}_2$ are in W_1 and W_2, respectively. If $\mathcal{M}_1$, that currently has high certainty, is refuted by the oracle set $\tilde{O}_p$, which is newly obtained from him, to the problem that naturally belongs to W_1, or if $\mathcal{M}_1$ can not cover a certain goal in $\tilde{O}_p$, then the system hesitates to dismiss $\mathcal{M}_1$ and tries to interpret the status of his conceptual discrimination as undifferentiated. If $\mathcal{M}_2$ satisfies $\tilde{O}_p$, $\tilde{O}_p$ is moved into W_2 and supports $\mathcal{M}_2$ there. Both discrimination conditions C_1 and C_2, which are world predicates of W_1 and W_2, are revised to be $C_1 \vee C_2$. If any clauses in any other worlds in the tree except the "another world" do not support $\tilde{O}_p$, it is assumed that he was thinking in the world that contains naive buggy knowledge and has solved the problem informally. $\tilde{O}_p$ is moved into the "another world" which is prepared to cover his unformulated problem solving, if one of the buggy clauses, prepared in the world in advance, satisfies $\tilde{O}_p$.

In our example, the student correctly answered several questions including Question 1 in the past, so that the student model had the correct clause to get the force which an object in uniformly accelerated motion receives. He made the wrong answer utilizing his naive physical world to Question 3 later. HSMIS receives oracles made from his answer, e.g., oracle(subtract(19.6,0,19.6),true), oracle(subtract(2,0,2),true). The above clause in the world of uniformly accelerated motion does not satisfy these oracles and derives an answer different from that of him. HSMIS recognizes the need for a new clause whose head is get_direction_of_force to satisfy the oracles. There exists the correct clause, however, which should have been applied to the problem, in the world of uniformly accelerated motion in the student model. The clause is reliable enough because the student correctly answered some questions using the method corresponding to the clause. The student, however, does not apply the method to the latest question. Therefore, it is unreasonable to assume he solved the problem in this world.

For this reason, THEMIS does not choose to resolve the inconsistency in the single world. Instead, it considers the situation as a multi-world contradiction and it searches for the world that already contains the clause which satisfies the oracles. If the search fails, THEMIS tests "another world" which represents students' naive problem solving. In this example, the system finds out the clause that represents his erroneous "motion implies force" misconception prepared in the naive world in advance. The system revises discrimination conditions that partition the concept of motion into the world of formulated physics and that of naive physics. It can explain his discrimination status, that he unstably applies physics formulas and naive knowledge.

4.3 Correcting student knowledge contradictions

By modeling student knowledge contradictions, the system is able to realize very effective and "real" Socratic tutoring as follows:

1. Give him a problem such that he tries to solve it in his naive world and fails to get a correct solution.
2. Remind him of the correct answer to the problem he obtained in Question 1.
3. Point out the inconsistency between the two results.
4. Explain the causes and guide him to build a correct concept.

In this way, he can correctly identify attributes necessary for building the concept, which we call world predicates, and establish relationships between them. Thus he can appropriately conceptualize the knowledge in both worlds. Obtaining the student model that represents the student knowledge contradiction, the system becomes able to generate an effective tutoring dialogue as in Figure 11.

The mechanism for single world contradictions explained earlier and the framework to handle multi-world contradictions explained in Section 4 are integrated into THEMIS. It gives up constructing a unified consistent model and dares to build a model in multiple worlds, when newly obtained data deny the reliable current model. The control mechanism guarantees the fidelity and the accuracy of

Tutor: Solve this problem. (Give a problem similar to Question 1 again)
The student correctly answers.
Tutor: You answered in Question 3 that the direction of the force which the sphere receives is upward at t=1s, because it is still moving upwards then. If that was correct, why didn't you say that the sphere in Question 1 receives force to the right?
Student: Because it was moving upwards, so it is hardly possible that it continued receiving force downward.
Tutor: The two problems are completely the same, e.g., speed at each time, etc., except for the direction of motion. If direction of force were to be implied from motion, you should have naturally said that the direction of the force is to the left in Question 1, but you didn't. (You should have ''motion implies force'' misconception in your naive physical world. $\cdots$)

Figure 11: A hypothetical example of tutoring behavior using student knowledge contradiction.

the model in each world. Student's incorrect inference methods, such as the use of abduction in his problem solving, can be formulated as contradictions. However, this topic has been kept for future work.

5 Discussions

Here some basic issues of comparative student modeling methodologies are discussed.

One cannot model students without defining a "bug" which is one of the key concepts of student modeling. The definition of bugs in turn defines the search space for student modeling. Bugs are defined as a *mal-function* or a *mal-structure* of the correct knowledge from the computational point of view. By mal-function and mal-structure, we mean a state in which a component of knowledge does not realize its function and a state in which the structure of knowledge is incorrect, i.e., some component is missing or inserted, etc., respectively. ¿From this point of view, there are three types of model which model:

1. what component of the correct knowledge is incorrect (mal-function-1)
2. how the component is incorrect (mal-function-2)
3. how the structure of the knowledge is incorrect (mal-structure)

Type 2 methods not only identify what component of the correct knowledge is incorrect in the student's head but also model how it is incorrect in a limited way, e.g., using fault models of each component. It cannot model, however, why the function is so incorrect. Needless to say, type 3 is the most powerful type of modeling, since it models what, how, and why the component is incorrect by modeling the incorrect structure of the knowledge.

Although student modeling is generally viewed as an inductive process in which a representation explaining observed data is built, we can find another view, i.e., as the analysis of the expertise model using the observed data (Hoppe, 1994). Modeling methods which obtain information of the student's understanding state by analyzing the correct knowledge based on the student behavior are called "analytic methods" here. A student model does not have to explain all the behavior of the student. Considering that its function is to give necessary and sufficient information to the tutoring module, analytic methods work very well for many types of tutoring modules. Typical examples of analytic methods are the overlay model (Carr *et al*, 1977), the logic programming method (Hoppe, 1994), and model-based cognitive diagnosis (Self, 1993a). This type of methods does not model mal-structure, while inductive methods try to represent it. Typical examples of inductive methods are the buggy model (Burton, 1982), perturbation methods (Otsuki *et al*, 1985), ACM (Langley *et al*, 1984), and THEMIS. Table 4 summarizes the characteristics of the methods discussed in this section.

Analytic methods

The Overlay model (Carr *et al*, 1977) is relatively easily used to control the modeling processes. That is why the overlay model has been used in many ITSs. Since it cannot represent the student's incorrect knowledge, the performance of overlay-model-based ITSs is limited.

Hoppe (1991; 1994) proposes a method to analyze the correct knowledge represented as a Prolog program by executing the program with the student answer as a goal. When the student answer is

Table 4: Evaluation of student models

System name	Analytic/ inductive	Mal-function-1	Mal-function-2	Mal-structure	Inconsistency	Theoretical foundation	Question automatically
Overlay	analytic	yes	no	no	no	low	NA
(Hoppe, 1994)	analytic	yes	yes	no	no	middle	NA
(Self, 1993a)	analytic	yes	yes	no	no	high	yes
IDEBUGGY	inductive	yes	yes	yes	partly	low	yes
Perturbation	inductive	yes	yes	yes	no	low	no
ACM/DPF	inductive	yes	yes	yes	partly	middle	no
THEMIS	*inductive*	*yes*	*yes*	*yes*	*yes*	*high*	*yes*

incorrect, the execution fails if it is interpreted by a Prolog interpreter. In order to recover the execution failure, he introduces a fail-safe meta-interpreter of Prolog and detects uncovered goals which potentially correspond to the student's bug. Many uncovered goals are usually detected, so he employs the EBG (Explanation-Based Generalization) technique to formulate rules called error patterns which are used to select the goals corresponding to the plausible bug. The search space of this method is small, since it is restricted to the computation tree of the correct knowledge. This method requires pre-defined classification rules which roughly correspond to bug rules, though they are more abstract than bug rules. It represents bugs of type mal-function-2. This method does not suffer from the inconsistency problem, because it builds a model from one datum.

Self (1993a) gives a characterization of student modeling as model-based cognitive diagnosis, making an application of GDE (deKleer *et al*, 1987). GDE is essentially based on an exhaustive search over the search space defined as a set of all the combinations of possible faulty components with the aid of ATMS. The search mechanism, in principle, is hence to check what set of components are faulty and to determine a minimum set of faulty components. Another advantage of GDE is its automatic question generation mechanism for disambiguation of the fault hypotheses. It represents only mal-function-1, although it does not require any bug library. Just a small extension, i.e., the introduction of fault modes into each component, enables it to cover type mal-function-2 as Self indicates. It is sensitive to inconsistent data. The theoretical foundation becomes more firm and sound in this order, though there is a large gap between the overlay model and Hoppe's method.

Inductive methods

IDEBUGGY (Burton, 1982) has many modular chunks of buggy procedures. It has the capability to cope with the noise problem, however, it is not complete. Its searching strategies depend on the assumption that each individual buggy procedure can be extracted from a combined buggy procedure. The burden of cataloging bugs is still left.

Bugs are viewed as variants of correct knowledge. They can be generated by applying perturbation operators to correct procedures. Perturbation methods are based on this idea. Takeuchi and Otsuki (1987) utilize this type of method in their system BOOK (Otsuki *et al*, 1985) in which two types of perturbation operators (domain-dependent and domain-independent ones) are introduced to augment the modeling power. The search space of this method is dependent on the complexity of the perturbation operators used, which is not usually so large. The method has no firm theoretical foundation for it but does not suffer from inconsistency of the data, since it builds a model from one observation.

Both ACM (Langley *et al*, 1984) and the authors' method THEMIS/HSMIS are based on the idea that student modeling is viewed as inductive learning from a set of examples. These two methods are almost equivalent in representation power of the student model. They can model not only mal-functions but *mal-structures* in terms of pre-defined primitives. Both methods act as domain-independent engines and their capabilities contain those of both the overlay and buggy methods. Besides, both of them can cope with noisy data. In general, it seems that the method of inference in ACM is not so sophisticated as that of HSMIS.

The major difference is their searching and diagnostic strategies for ascertaining what part of the model conflicts with the student. HSMIS can prune the search space for new clauses in a refinement graph, although ACM originally made a blind search of all its search space. To embody efficient searching, Langley *et al* (1990) have been developing DPF (Dynamic Path Finder).

HSMIS can incrementally revise the model using newly obtained data. Although the path-finding

algorithm is formulated by DPF, the rule-finding process is not formally defined. HSMIS can distinguish between noisy data and bug migration (Kono *et al*, 1993a; Ikeda *et al*, 1993), while ACM cannot currently (Langley *et al*, 1990). HSMIS asks questions which are logically required in consideration of educational appropriateness, while ACM does not.

Other types of methods

Huang (1993) proposes a logic to capture the student's inconsistent knowledge. Surprisingly, only his study, apart from THEMIS, has so far been carried out on modeling student knowledge contradictions. Although it is well-defined in terms of propositional calculus, it would be difficult to extend it to deal with first order predicate calculus.

The role of a student model is to provide a tutoring module with information necessary for generating adaptive behavior. There is no need to produce precise information more than required. PROTO-TEG (Dillenbourg, 1989) is an interesting ITS based on this idea. It does not generate any structured information that can be regarded as a student model, but generates heuristics to produce tutoring behavior directly by incorporating one of the inductive learning theories, LEX (Mitchel *et al*, 1984). It is successful in handling its simple didactic strategies.

Considering the functions the student model should perform and the intractability of modeling, Ohlsson (1993) proposes a new approach to student modeling called constraint-based in which domain knowledge is represented by a set of constraints and the student model is represented by constraint violation. The idea seems good, since it can avoid many of the difficulties which current modeling technologies are suffering from. One has to identify, however, sets of constraints which cover all the information the tutoring module requires which is not an easy task.

6 Concluding remarks

This paper has presented a comprehensive student modeling methodology and its use in an ITS. Contradictions we have to cope with during modeling students are first defined. As a result, we obtained four types of contradiction, including contradictory knowledge possessed by students. The modeling system of such students has to model undifferentiated concepts and inconsistency as it is. A sophisticated control mechanism to deal with both single world and multi-world contradictions for THEMIS has been developed.

Finally, THEMIS has been compared with other modeling systems from various viewpoints to demonstrate that it is a well-defined and generic student modeling algorithm, which can build a student model of high representation power. THEMIS has been implemented in Common-ESP(Extended Self-contained Prolog) (AIR, 1990), and HSMIS is embedded in FITS, Framework for ITS. Two ITSs have been built using FITS, one on geography and the other on chemical reactions (Mizoguchi *et al*, 1987; Mizoguchi *et al*, 1991; Ikeda *et al*, 1994).

References

AI Language Research Institute (ed). (1990). *CESP Language Guide*, 5-1-1, Ohuna, Kamakura, Kanagawa 247, Japan.

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

Carr, B. & Goldstein, I. (1977). Overlays:A theory of modeling for computer aided instruction, *MIT AI Memo 406*.

Clement, J. (1982). Students' preconceptions in introductory mechanics, *American Journal of Physics*, **50**, 66–71.

de Kleer, J. (1986). An assumption-based TMS, *Artificial Intelligence*, **28**, 127–162.

de Kleer, J. & Williams, B. C. (1987). Diagnosing multiple faults, *Artificial Intelligence*, **32**, 97–130.

Dillenbourg, P. (1989). Designing a self-improving tutor: PROTO-TEG, *Instructional Science*, **18**, 193–216.

Dillenbourg, P. & Self, J. (1992). A framework for learner modelling, *Interactive Learning Environments*, **2**(2), 111–137.

Hoppe, H. U. (1991). An analysis of EBG and its relation to partial evaluation: lessons learned, *Arbeitspapiere der GMD*.

Hoppe, H. U. (1994). Deductive error diagnosis and inductive error generation for intelligent tutoring systems, *Journal of Artificial Intelligence in Education*, (to appear).

Huang, X., McCalla, G. I., Greer, J. E. & Neufeld, E. (1991a). Revising deductive knowledge and stereotypical knowledge in a student model, *User Modeling and User-Adapted Interaction*, **1**, 87–115.

Huang, X., McCalla, G.I. & Neufeld, E. (1991b). Using attention in belief revision. In *Proc. AAAI-91*, 275–280.

Huang, X. (1993). Inconsistent beliefs, attention, and student modeling, *Journal of Artificial Intelligence in Education*, **3**(4), 417–428.

Ikeda, M., Mizoguchi, R. & Kakusho, O. (1988). A hypothetical model inference system, *Trans. of IEICE Japan*, **J71-D**, 1761–1771 (in Japanese).

Ikeda, M., Mizoguchi, R. & Kakusho, O. (1989). Student model description language SMDL and student model inference system SMIS, *Trans. of IEICE Japan*, **J72-D-II**, 112–120 (in Japanese).

Ikeda, M., Kono, Y. & Mizoguchi, R. (1993). Nonmonotonic model inference -A formalization of student modeling-, In *Proc. IJCAI'93*, Chambery, France, 467-473.

Ikeda, M. & Mizoguchi, R. (1994). FITS: A framework for ITS – A computational model of tutoring –, *Journal of Artificial Intelligence in Education*, **5**, (to appear).

Kawai, K., Mizoguchi, R., Kakusho, O. & Toyoda, J. (1987). A framework for ICAI system based on inductive inference and logic programming, *New Generation Computing*, **5**(1), 115–129.

Kono, Y., Ikeda, M. & Mizoguchi, R. (1992). To contradict is human -Student modeling of inconsistency-, In Frasson, C., et al. (eds.), *Intelligent Tutoring Systems*, ITS'92 Proceedings, 451–458, Springer-Verlag.

Kono, Y., Tokimori, T., Ikeda, M., Nomura, Y. & Mizoguchi, R. (1993a). A student model building method based on formalization of nonmonotonicity, *Journal of Japanese Society for Artificial Intelligence*, **8**(4), 488–498, (in Japanese).

Kono, Y., Ikeda, M. & Mizoguchi, R. (1993b). A modeling method for students with contradictions, In *Proc. AI-ED 93*, Edinburgh, Scotland, 481–488.

Kono, Y., Ikeda, M. & Mizoguchi, R. (1994). An inductive student modeling method which deals with student contradictions, *IEICE Trans. on Information and Systems*, **E77-D**(1), 39-48.

Langley, P. & Ohlsson, S. (1984). Automated cognitive modeling, In *Proc. AAAI-84*, 193–197.

Langley, P., Wogulis, J. & Ohlsson, S. (1990). Rules and principles in cognitive diagnosis, In Frederiksen, N., et al. (eds.), *Diagnostic Monitoring of Skill and Knowledge Acquisition*, 217–250, Lawrence Erlbaum.

Mitchel, T., Utgoff, P. & Benerji, R. (1984). Learning by experimentation: Acquiring and refining problem solving heuristics, In Michalski, R., et al. (eds.), *Machine Learning*, Springer-Verlag.

Mizoguchi, R., Ikeda, M. & Kakusho, O. (1987). An innovative framework for intelligent tutoring systems, In *Proc. IFIP TC3 Working Conference on AI Tools in Education*, 105–120, Fascati, Italy.

Mizoguchi, R. & Ikeda, M. (1991). A generic framework for ITS and its evaluation, In Lewis, R. & Otsuki, S. (eds.), *Advanced Research on Computers in Education*, 63–72, North-Holland.

Ohlsson, S. (1993). Constraint-based student modeling, *Journal of Artificial Intelligence in Education*, **3**(4), 429–447.

Otsuki, S. & Takeuchi, A. (1985). Intelligent CAI system based on teaching strategy and learner model, In *Proc. WCCE 85*, 463–468.

Self, J. (1988). Bypassing the intractable problem of student modelling, In *Proc. ITS'88*, 18–24, Montréal, Canada.

Self, J. (1993a). Model-based cognitive diagnosis, *User Modeling and User-Adapted Interaction*, **3**, 89–106.

Self, J. (1993b). Formal approaches to student modeling, In McCalla, G.I. & Greer, J. (eds.), *Student Modelling*, Springer-Verlag.

Shapiro, E. Y. (1981). *Inductive Inference of Theories from Facts*, *Yale University Research Report 192*.

Shapiro, E. Y. (1982). *Algorithmic Program Debugging*, MIT Press.

Stevens, A. L., & Collins, A. (1977). The goal structure of a Socratic tutor, *Proc. National ACM Conference*, 256–263.

Takeuchi, A. & Otsuki, S. (1987). Formation of learner model by perturbation method and teaching knowledge, *Trans. Information Processing Society of Japan*, **28**(1), 54–63, (in Japanese).

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers.

Woolf, B. P. & Murray, T. (1993). Using machine learning to advise a student model, *Journal of Artificial Intelligence in Education*, **3**(4), 401–416.

Acknowledgments

The authors would like to thank Takeo Tokimori for his contribution to this work. The authors are grateful to Geoffry Webb for his comment which motivated the research on student contradictions. The authors are also thankful to Heinz Ulrich Hoppe for his valuable comments. The authors greatly appreciate the reviewers' precise and valuable comments. This work is supported in part by Grant-in Aid for Scientific Research on Priority Areas of the Ministry of Education, Science and Culture of Japan under Grant: (No. 03245106).

Appendix

A Formal definition of execution of SMDL program

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: H::-$B_1, B_2, \cdots, B_n$, $n \geq 0$, where $H, B_i (1 \leq i \leq n)$ are predicates. In the case of $n = 0$, it is called *a fact*. The SMDL program P is a finite set of clauses.

[Definition A.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 resolved from the goal G and the clause C with θ. When $k \geq 1$, $\theta = \{T/(T_1 \wedge \cdots \wedge T_k)\} \cup \theta_0$. This operation is called *weak resolution* and denoted by a triple $< G, \theta, C >$. $G\theta_0\theta_1 \cdots \theta_{l-1}$ is *weakly-resolved* from P, if there exists a sequence of the weak resolution: $< G, \theta_0, C_0 >, < G_1, \theta_1, C_1 >, \cdots, < \phi, \theta_l, C_l >$, where G_i is a goal weakly-derived from G_{i-1} and C_{i-1} with θ_{i-1}, $C_i \in P\,(0 \leq i \leq l-1)$. □

[Definition A.2] For a given goal $G = p(\tilde{X}', T')$, if there exist a maximal 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 resolved from G and S with θ_0.

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

This operation is called *strong resolution* and denoted by a triple $\ll G, \theta_0, S \gg$. $G\theta_0, \theta_1, \cdots, \theta_{l-1}$ is said to be *strongly-resolved* from P, if there exists a sequence of the strong resolution 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 G_i is a goal strongly-derived from G_{i-1} and S with θ_{i-1}, and $S_r \subseteq P (0 \leq r \leq l-1)$. □

[Definition A.3] When a goal $G = p(\tilde{X}', T')$ and a SMDL program P are given, the result of resolution 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. □

B Formal definition of terms for SMIS

[Definition B.1] 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 $T' \in \{true, false, unknown\}$. □

[Definition B.2] The clause $C = p(\tilde{X}, T)$::-$q_1(\tilde{X}_1, T_1), q_2(\tilde{X}_2, T_2), \cdots, q_k(\tilde{X}_k, T_k) \in P$ is said to cover $p(\tilde{X}', T')$, 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 $\{< p(\tilde{X}', T), T' >\} \cup \{< q_i(\tilde{X}'_i, T_i), T'_i > | 1 \leq i \leq k\} \subset \Omega$. The $q_1(\tilde{X}'_1, T'_1), \cdots, q_k(\tilde{X}'_k, T'_k)$ is called a *top-level trace* of C for $p(\tilde{X}', T')$. □

[Definition B.3] The model P is said to be *too 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. □

[Definition B.4] $T_1 \succ T_2$ iff $T_1 = T_1 \vee T_2$, where $T_1, T_2 \in \{true, false, unknown\}$ and $T_1 \neq T_2$. □

[Definition B.5] Assume that $< p(\tilde{X}', T), T' > \in \Omega$ and $T' \in \{unknown, false\}$. The model P is said to be *too strong* if $p(\tilde{X}', Tg)$ such that $Tg \succ T'$ is weakly-derived from P. □

[Definition B.6] 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 Ω-*refuted*, 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 Ω-*refutation* for C. □

[Definition B.7] 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 □

C ATMS

C.1 Data structure of ATMS

ATMS records a datum in the following form,

$$< D, \{E_1, E_2, \cdots, E_k\}, \{J_1, J_2, \cdots, J_m\} >$$

where D is the datum used in the reasoning system. E_i is an environment where D is derived and the set $\{E_i | 1 \leq i \leq k\}$ is called a label of D. J_i is a justification informed of by the reasoning system. The data recorded in this form is called an ATMS node, which is classified into the 3 type, i.e. a premise node, an assumed node, and a derived node.

The premise node includes an empty environment in the label: $< D, \{\{\}, \cdots\}, \{\cdots\} >$. The assumed node has the same set composed of a single identifier as a label and justification: $< D, \{\{a_i\}\}, \{\{a_i\}, \cdots\} >$, where a_i is a identifier of assumption (in this paper, the identifier of the assumption will be expressed with the suffixed letter a). Nodes except premise nodes and assumption nodes are derived nodes. ATMS updates the labels of the individual nodes, based on the justifications informed from the reasoning system.

C.2 Detection and resolution of contradiction

A situation in the problem solving process is called a context, which is defined with a set of data hold on the situation. An environment deriving all the data included in the context is called a characteristic environment of the context.

When the contradicted data $D_1, D_2, ..., D_n$ are detected in a context, the reasoning system informs it to ATMS in the following form.

$$D_1, D_2, \cdots, D_n \Rightarrow \bot$$

When derivation of inconsistency is informed, ATMS calculates and records the label of $\bot$. The reasoning system ceases to solve the problem in the contradicted context and transfer to a new consistent characteristic environment.

With regard to the nodes that have been derived until that time, ATMS determines whether the nodes holds (*in*) or does not hold (*out*) in the new characteristic environment in accordance with the conditions shown below. Thus a new context is composed with a set of *in* nodes.

[Condition 1] Let the label of the node n be L and the characteristic environment be C_e.

if $\exists E(E \in L$ **and** $E \subset C_e)$
then status of n := *in*
else status of n := *out* □

D Formulation of the inference process of THEMIS

D.1 Description of model inference process of SMIS

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

if cond1: $(< p(\tilde{X}',T),T' > \in \tilde{\Omega}\,)$ **and**
cond2: (C is Ω-consistent) **and**
cond3: (C covers $p(\tilde{X}',T')$) **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.

$$< \mathbf{oracle}(p(\tilde{X}', T), T'), \{\{a_i\}\}, \{\{a_i\}\} > [3]$$

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

$$< \Omega\text{-}\mathbf{consistent}(C), \{\{a_i\}\}, \{\{a_i\}\} >$$

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.

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

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 $p(\tilde{X}', T')$.

ATMS produces a node in the following form.

$$< cover(C, A'), \{\cdots\}, \{\{\mathbf{oracle}(p(\tilde{X}', T), T'), \\ \mathbf{oracle}(q_1(\tilde{X}_1, T_1)\theta, T'_1), \cdots, \mathbf{oracle}(q_k(\tilde{X}_k, T_k)\theta, T'_k)\}\} >$$

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.

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

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

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

ATMS generates the following ATMS node based on the justification.

$$< model(C), \{\cdots\}, \\ \{\{\mathbf{oracle}(p(\tilde{X}', T), T'), \Omega\text{-}\mathbf{consistent}(C), \mathbf{general}(C), cover(C, p(\tilde{X}', T'))\} >$$

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.

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

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.

ATMS produces the following ATMS node.

[3] 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 datum 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).

$< \Omega\text{-}refutation(C), \{\cdots\},$
$\{\{\mathbf{oracle}(p(\tilde{X}', T), T'),\ \mathbf{oracle}(q_1(\tilde{X}'_1, T_1)\ , T'_1), \cdots, \mathbf{oracle}(q_k(\tilde{X}'_k, T_k), T'_k)\}\} >$

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

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

D.2 Controlling mechanisms of the modeling process

D.2.1 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 knowledge or plausible buggy knowledge 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 assumption $trust(C)$, 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:

$$\left.\begin{array}{l}\mathbf{trust}(C)\\ \mathbf{oracle}(p(\tilde{X}', T), T')\end{array}\right\} \Rightarrow v\_oracle(q_i(\tilde{X}'_i, T_i), T'_i))$$

SMIS treats oracles and *v_oracles* in the same manner.

D.2.2 Meta-Oracles

When the clause C is added to the model based on a "meta-oracle", HSMIS informs ATMS of the following justification.

$$\left.\begin{array}{l}\mathbf{metaOracle}(C, yes)\\ \Omega\text{-}\mathbf{consistent}(C)\\ \mathbf{general}(C)\end{array}\right\} \Rightarrow model(C)$$

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

$$\mathbf{metaOracle}(C, no) \Rightarrow \Omega\text{-}refutation(C)$$

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

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

D.3 Formulation for multi-world model inference

Here explains the extended formulation of modeling process in THEMIS.

A set of instantiations for the input arguments of the top level goal is called a problem. When the system assumes that the student understands that a problem $\tilde{P}$ belongs to a certain world $\mathcal{W}_1$, ATMS is informed of the following assumption:

$$< \mathbf{belong}(\tilde{P}, \mathcal{W}_1), \{\{a_i\}\}, \{\{a_i\}\} >$$

A clause which is added to the model should be dealt with separately in each world. Thus, each ATMS node "model" has an additional justified assumption "belong" in its each justification.

$$\left.\begin{array}{l}\mathbf{oracle}(p(\tilde{X}', T), T')\\ \Omega\text{-}\mathbf{consistent}(C)\\ \mathbf{general}(C)\\ \mathbf{belong}(\tilde{P}, \mathcal{W}_1)\\ cover(C, p(\tilde{X}', T'))\end{array}\right\} \Rightarrow model(C, \mathcal{W}_1) \qquad \left.\begin{array}{l}\mathbf{metaOracle}(C, yes)\\ \Omega\text{-}\mathbf{consistent}(C)\\ \mathbf{general}(C)\\ \mathbf{belong}(\tilde{P}, \mathcal{W}_1)\end{array}\right\} \Rightarrow model(C, \mathcal{W}_1)$$

When it is found that the student applied clauses that belong to $\mathcal{W}_2$ to $\tilde{P}$ which should have been solved in $\mathcal{W}_1$, ATMS is informed of it and it generates an assumption $\mathbf{belong}(\tilde{P}, \mathcal{W}_2)$, and further ATMS is informed of that the student does not yet discriminate between $\mathcal{W}_1$ and $\mathcal{W}_2$ with the following justification.

$$\mathbf{belong}(\tilde{P}, \mathcal{W}_2) \Rightarrow indiscriminate(\mathcal{W}_1, \mathcal{W}_2)$$

It is assumed in advance that the student correctly discriminates a certain pair of worlds:

$$< \mathbf{discriminate}(\mathcal{W}_1, \mathcal{W}_2), \{\{a_i\}\}, \{\{a_i\}\} >$$

Assume that the system is going to give the student a problem $\tilde{P}$ that belongs to the world $\mathcal{W}_1$. If $\mathbf{discriminate}(\mathcal{W}_1, \mathcal{W}_2)$ is *in*, The student is expected to solve $\tilde{P}$ in $\mathcal{W}_1$ and ATMS is informed of it in the following form.

$$\left.\begin{array}{l}\mathbf{belong}(\tilde{P}, \mathcal{W}_1) \\ \mathbf{discriminate}(\mathcal{W}_1, \mathcal{W}_2)\end{array}\right\} \Rightarrow solveIn(\tilde{P}, \mathcal{W}_1)$$

When an ATMS node $solveIn(\tilde{P}, \mathcal{W}_1)$ is derived and is *in* , and a reliable clause C in the model which belongs to $\mathcal{W}_1$ is unifiable with $\tilde{P}$, the system predicts student's answer, i.e., an *oracle*, and ATMS node $modelPrediction(\tilde{P}')$ is derived in the following form.

$$\left.\begin{array}{l}solveIn(\tilde{P}, \mathcal{W}_1) \\ model(C, \mathcal{W}_1)\end{array}\right\} \Rightarrow modelPrediction(\tilde{P}')$$

D.4 Detection of contradiction

The contradiction derived in the inference process of HSMIS is classified into the following ten types.

S1) Contradiction of Ω-consistency: When a refutation for the clause, whose correctness has been assumed, is detected due to appearance of new oracles, the contradiction is detected by the following rule.
Ω-$\mathbf{consistent}(C)$ & Ω-$refutation(C) \Rightarrow \bot$

S2) Contradiction of cover test: When a correct top level trace is found for the clause due to appearance of new oracles, the contradiction is detected by the following rule.
$\mathbf{uncover}(C, A)$ & $cover(C, A) \Rightarrow \bot$

S3) Contradiction of generality: When both of clauses C and C', which suffices the relation $C \xrightarrow{\rho} C'$, are in the current model, the inconsistency is detected by the following rule.
$\mathbf{general}(C')$ & $model(C) \Rightarrow \bot$

S4) Contradiction of trust: $\mathbf{trust}(C)$ & Ω-$refutation(C) \Rightarrow \bot$

S5) Contradiction of meta-oracle: $\mathbf{metaOracle}(C, yes)$ & Ω-$refutation(C) \Rightarrow \bot$

S6) Contradiction of oracle: When $< A, T_1 > \in \tilde{\Omega}$, $< A, T_2 > \in \tilde{\Omega}$ and $T_1 \neq T_2$, the oracle environment is evidently inconsistent. The inconsistency detecting rule is as follow.
$\mathbf{oracle}(A, T_1')$ & $\mathbf{oracle}(A, T_2')$ & $T_1' \neq T_2' \Rightarrow \bot$

S7) Failure of search: When search for a clause to cover an uncovered goal turns out a failure, the failure is dealt with as contradiction. For each clause C_i $(1 \leq i \leq m)$ searched for to cover $< A, T' >$: $\&_{i=1}^{m}($ $\mathbf{uncover}(C_i, A')$ or Ω-$refutation(C_i)) \Rightarrow \bot$

M1) Contradiction of belong: $\mathbf{belong}(\tilde{P}, \mathcal{W}_1)$ & $\mathbf{belong}(\tilde{P}, \mathcal{W}_2)$ & $\mathcal{W}_1 \neq \mathcal{W}_2 \Rightarrow \bot$

M2) Contradiction of discrimination: $\mathbf{discriminate}(\mathcal{W}_1, \mathcal{W}_2)$ & $indiscriminate(\mathcal{W}_1, \mathcal{W}_2) \Rightarrow \bot$

M3) Contradiction of model prediction and oracle: $\mathbf{oracle}(\tilde{P}_0)$ & $modelPrediction(\tilde{P}')$ & $\tilde{P}_0 \neq \tilde{P}' \Rightarrow \bot$

When any types of the contradiction is detected, ATMS is informed of it and updates the nogood record.